Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable 8.1
Definition of Service Provisioning Goals, Economic Impacts, and SLA Management Tasks

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universidat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM/UniBwM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UPI, Romania

For more information on this document or the EMANICS project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France

Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Definition of Service Provisioning Goals,
Economic Impacts, and SLA Management Tasks

Type: Public

Editors: Burkhard Stiller, David Hausheer, Gregor Schaffrath

E-mail: stiller@ifi.unizh.ch, hausheer@ifi.unizh.ch, schaffrath@ifi.unizh.ch

Authors: (alphabetic order) Mark Burgess, Gabi Dreo Rodosek, Stylianos Georgoulas, Matthias Göhner, David Hausheer, Aiko Pras, Thomas Schaaf, Burkhard Stiller, Martin Waldburger, Ning Wang

Doc ID: D8.1-v2.0

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1-0.92	May 30, 2006	Complete list of detailed author contributions available in D8.1-v0.92	Complete change history available in D8.1-v1.0
V1.0	May 30, 2005	Burkhard Stiller	Editing of multiple intro-sections, references, abbreviations
V1.1-1.9	June 23, 2006	Complete list of detailed author contributions available in D8.1-v1.9	Complete change history available in D8.1-v1.9
V 2.0	June 30, 2006	Burkhard Stiller	Including final comments and updates received from David Hausheer,

Table of Contents

1 Executive Summary

Managing the heterogeneous and partly broadband-capable infrastructure of tomorrow's Internet requires suitable and scalable technology as well as mechanisms in support of viable economic means of service deployment and provisioning. Such services will advance beyond the pure Internet Protocol (IP) access and need to cover value-added services as well. Currently, an economic unfairness exists in the current Internet and in a converged IP infrastructure also.

Thus, work in Work Package 8 (WP8) so far has been documented in this deliverable D8.1, addressing readers being familiar with traditional network management and basic economics. The state-of-the-art on both areas, focussing mainly on the use case of network service provisioning, has been pin-pointed and due to the presentation of newly developed, integrated concepts as well as ideas the discussion of emerging, urgent, and required research questions, technology requirements, and future economic management schemes has been included.

Driven by these considerations, the need for developing advanced network management techniques becomes obvious, of course, only based on well defined roles and models, covering promises and interactions, that will enable multiple parties to access this infrastructure in a technically suitable, efficient, and economically fair manner. This step beyond the current state of network management — in terms of models and mechanisms — will lead to a common and administrable information infrastructure, which in turn can be operated economically and efficiently under predefined assumptions.

Addressing those integrative trends does require to define and specify models, which will follow the key concerns of technology and economic mechanism requirements. Thus, the set of five models forms the basis for an integrated network management modelling, addressing economic and technical aspects in an integrated manner. The role model determines players together with their specific behavior in the multi-user and multi-provider domains of service usage and offering. The service model outlines the type of services envisioned and specified. The charging model identifies key activities to be undertaken for being able to pay for a certain service usage a fair and announced price. Finally, business models, as shaped by those three models mentioned, find instantiations for operations in deployment models.

Based on this set of basic models, a number of key mechanisms — in support of technical and economic dimensions — and, thus, in support of an overall economic management are analyzed, defined, and discussed in a close level of detail. Since mechanisms in support of management tasks in a distributed system consist out of the two different types of technical mechanisms and economic mechanisms, the approach with Service Level Agreements (SLA) defines service quality and delivery actions, which specify services and their characteristics. The subsequent service provisioning process — driven by those specifications and guided by the service model's interdependencies of service functionality as well as network management support functionality — ensures a customer/user to provider relation based in common grounds. In addition, this specification is followed by a negotiation scheme, under which those two roles will be able to agree on parameters, parameter sets, values, and characteristics to be applied explicitly in a given service usage. This overall process may be guided by policies, determining rules, which do address context-specific details to be applied for a certain case, such as operator specifics, service characteristics, or market segment concerns. Finally, the accounting in the technical sense addresses the collection of data, which originate from measurement points and define the

service-specific definitions of resources, actions, and events, which have taken place in the service usage phase. Those mechanisms will serve as the basis for the charge calculation based on any type of accounted for information, which combine technical and economic parameters to ensure that the business model is attached to the service model via the charging model and vice versa.

To be able to address those economic basics in the most fine granular level of details possible and for defining a suitable service quality and delivery action, economic measures and cost parameters form the appropriate basics. This definition and a discussion of a sample is followed by a detailed view onto mobile micro-payment schemes and their respective accounting systems in support of extremely low-value transactions required for various on-line services. Furthermore, the analysis of two concrete examples is undertaken in terms of service provisioning templates, under the assumptions of existing SLAs.

Finally, work in WP8 went beyond a pure formal deliverable. Therefore, a number of joint papers of EMANICS partners investigates emerging research questions on economics in network management in a wider range, such as bandwidth trading, accounting for grids, uncertainty modelling, and Peer-to-peer (P2P) fairness management. Those to be worked on, intermediate, and already finalized results have been included as an Appendix to D8.1 to document the work undertaken and the collaboration results achieved.

2 Introduction

Managing the broadband-capable infrastructure of tomorrow's Internet — based on the Internet Protocol IP — requires suitable *technology and mechanisms* as well as valid and viable economic means, which will

- lead to a generic information infrastructure and
- enable multiple parties' access to this infrastructure in an economically fair manner.

In particular the *economic dimension of network management* has to be an integrated part of an overall broadband network solution, including existing technologies on layer 2 (Data Link Layer), such as POTS (Plain Old Telephony System), FR (Frame Relay), ISDN (Integrated Services Digital Network), WLAN (Wireless Local Area Network), and layer 1 (Physical Layer), such as SDH (Synchronous Digital Hierarchy) or xDSL (Type x Digital Subscriber Line). While different technical and economic measures depend on various factors and influence providers' charging models, their

- mechanisms for broadband accounting and network state supervision,
- their management optimization dimension, and
- their way of technical operations

have to be integrated in a common fashion.

Addressing especially the *economics of the Internet* as a whole, the Internet in its current form has determined the basis for many successful business models and companies in turn [36]. However, these companies typically belong to one or more of the following categories: equipment vendors, software vendors, advertisers, and retailers. In general, those successful companies did not belong to the carrier and IP service provider category [36]. Thus, three problems [36] may be considered to be one major root of the Internet profit dilemma:

- "The fixed-price model of Internet access, created when dial-up connections limited how much traffic a user could generate or receive, makes it hard to charge more for new uses of the Internet. This has encouraged exploitation and expansion of the Internet usage, but without a proportional increase in the revenues of those who must invest to expand the capacity of the Internet itself—and more capacity is needed to sustain more usage" [36]. The view on Internet pricing and communications in general may be obtained in a larger and complete overview in [38].
- "The lack of commercial-grade interfaces between Internet Service Providers (ISP)—i.e., interfaces that include guarantees of performance and settlement for service participation—has discouraged the deployment of enhanced features like Quality-of-Service (QoS). There are technical means to provide QoS between ISPs, but there exists no legal requirement for inter-ISP settlements, like that for most Public Switched Telephony Network (PSTN) common carriers. Additionally, no business-level solution from the Internet community seems to be in place in a agreeable manner" [36].
- "Adding capacity to an IP network means adding resources. While those resources can be offered for sale as "priority services" supporting a theoretical premium service class, they also generally make "best effort" services better, encouraging users to roll the dice on how good "best effort" might be, and making it harder to sell premium services" [36].

Additionally, the well-received notion of *convergence* — basically determining the integration of IP and telecommunications on a single platform — or even the adoption of an IP infrastructure as the universal communications basis for all services emphasized this dilemma. Due to these dramatic technology changes an additional area of problems has been created:

- "If convergence on IP means convergence on the Internet, revenues for today's legacy services seem doomed to disappear" [36].

Since these services in the U.S.A. currently constitute more than US$ 240 billion in annual revenues, and the U.S.A. market for Internet services at full business and consumer penetration (including broadband modernization) is probably worth no more than between US$ 50 billion and US$ 80 billion in annual revenue, this would mean killing more than two-thirds of the current service provider revenues [36].

These issues pinpoint the need of well defined management techniques that will lead to a generic information infrastructure and will enable multiple parties access to this infrastructure. Additionally, while major factors as of technical and economic management have been addressed until now in the research community as separated activities, such as achieving technical efficiency of accounting modules, definitions of charging models independent of technological prerequisites, maintaining viable Service Level Management (SLM) aspects, and determining management-independent broadband services models, the inter-operation model of those essential components still remains to be designed. This is one task of this WP8 of EMANICS.

The development of a customer service management approach and corresponding interfaces as the foundation of the integration of a production environment is essential. In addition, the mapping of different QoS architectures and parameters between organizational boundaries and platforms will lead to an approach to map QoS parameters to the infrastructure, across organizational boundaries, and onto charges to be paid for by the customer. The envisioned future of broadband networks has the potential to become an enabler for attractive new kinds of services and applications in the business as well as the private sectors. Therefore, the need for appealing new service offerings and their management solutions is emerging. Those solutions have to be provided to customers with a guaranteed QoS at an agreed upon or negotiated cost and have to be captured in a legally binding and technically supervised contract, typically combined into a Service Level Agreements (SLA), which includes the Service Level Specification (SLS).

These areas — amongst other to be described and analyzed — determine those collaborative and joint research efforts addressed in the initial form in this deliverable of EMANICS' WP8. Therefore, an investigation and integration effort to make those effects configurable to various network and service provider needs is targeted at within this work-package.

2.1 Purpose of the Document D8.1

Deliverable D8.1 addresses two key questions and two groups of readers. While the technical purpose of this document is to describe and document the state-of-the-art in advanced network management and economic control schemes, mainly focussing on the network service provisioning case, it summarizes additionally on a per-model and mechanisms basis the key problems to be addressed and solved in an integrated economic management approach. New and advanced possibilities are explored, which combine the traditionally separately considered domains of technical and economic management.

Thus, the readership envisioned covers on one hand EMANICS partners to obtain a clearly defined and well explained view on roles, models, and mechanisms, which form the basis for many steps undertaken in EMANICS. On the other hands, the document is addressed toward the network management community as a whole to document the advantages to be encountered, if economic management techniques are integrated fully and transparently into traditional models of network management.

2.2 Document Outline

The initial set of basic models (cf. Section 3) and key mechanisms (cf. Section 4) — addressing technical and economic dimensions — in support of an overall economic management are outlined below. Following a short section on preliminary conclusions in Section 5, a glossary of key terms in Section 6 summarizes key terminology. Bibliographic references in Section 7 and a list of abbreviations in Section 8 complement this work with respective data for further study or organizational help.

Finally, as it is intended for a Network of Excellence major selected cooperation work is applied on top of this fundamental basis work in terms of an initial set of joined papers, as attached in the appendices of Section 10. Apart from those, a limited number of single affiliation papers have been added, which started in the technical and economic areas of EMANICS and WP8 before the official start of the Network of Excellence and which underline as well as the strong research efforts being undertaken by WP8 partners.

3 Existing Key Models

The basis for any type of joint and collaborative work, especially in the field of network management, is laid by the definition of core models, which determine the key for common agreements, technologies, and mechanisms to be applied. Thus, based on the summary of key terms and their definitions, the following five models are introduced, their inter-relations are discussed, and their major basics are described:

- Deployment Model
- Business Model
- Role Model
- Service Model
- Charging Model

Those will be used within the Economic Management Work Package WP8 to base all joint work. Additionally, other EMANICS work-packages will benefit from many precise definitions, which determine the basic outline of technical and economic management foundations.

To provide an initial and structured overview of the approach undertaken here, Figure 1 summarizes those five major EMANICS models and outlines their course-grained interdependencies in terms of a three level hierarchy. Thus, Figure 1 includes the respective assignment of business model elements to the underlying models of roles, services, and charging. In addition, it depicts that the deployment model forms a formalized path to an instantiation of the business model based on a set of concrete and case-specific assumptions.

Accordingly, the EMANICS-specific perception of business models is determined by means of (a) which value propositions (Service Model; cf. Section 3.3) are made and (b) a single or multiple interacting actors characterized by player-specific incentives and behavior (Role Model; cf. Section 3.2) are able to achieve economic profits (Charging Model; cf. Section 3.4).

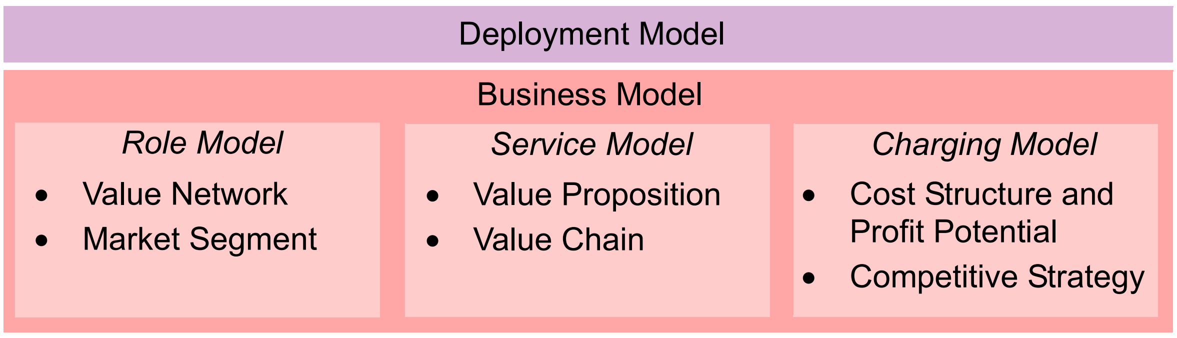

Figure 1: EMANICS Business Modeling Overview

3.1 Business Model

A vast number of business model definitions is found in economic literature. In the comprehensive study of [39], various existing approaches are outlined, mainly by comparing and classifying them according to various dimensions. This investigation was driven originally by a focus on business models for electronic commerce. Even though this narrow perspective was widened again in order to cover approaches not being Information and Communication Technology (ICT)-centered only, this study represents a highly valuable resource in terms of an embracing overview of business models applicable to electronic businesses, thus, being especially applicable to network management areas and technologies of electronic service provisioning.

From the wide range of approaches to business modeling, [6] determines the most relevant approach and it reveals the following major components typically to be present in business models:

- **Value Proposition:** The main objective for any business model is to determine how and by what means value creation is enabled in an organization. Value creation is particularly of interest in fully competitive environments, since it represents the steps in the overall value chain that can be exploited commercially. As with respect to the specific context of EMANICS, value is assumed to be created mainly by offering basic or complex, composed electronic services to the customer base. Thus, services are perceived as a form of electronic products, providing a certain amount of utility to the service user for which, in turn, the service customer will be charged.
- **Market Segment:** Services are designed in order satisfy customer needs. Depending on the specific market environments, both, service providers and service users may show different incentive settings which determine behavior patterns and, thus, limit the range of envisaged market segments. Accordingly, marketing activities in a Business-to-Business (B2B) context vary from the Business-to-Consumer (B2C) area. In particular, these differences bring about contractual agreements that are adapted to a given market segment. For EMANICS, Service Level Agreements (SLA) and Service Level Specifications (SLS) are of high interest in this context.
- **Value Chain:** In order to create value, usually several steps in an organization are needed. These steps are modeled by means of processes and workflows. Besides activities, also complementary assets have to be considered. As those procedures are assumed to create commercially exploitable value, their actual implementation encapsulates business-critical knowledge. This determines on the one hand and to a certain extent the competitive strategy an actor takes, on the other hand this knowledge will be protected from inspection by organization-external entities.

- **Expected Cost Structure and Profit Potential:** The basic economic principle consists in the rule to pursue activity A if, and only if, related costs do not exceed economic benefits of A. Costs include also opportunity costs, meaning costs of not pursuing alternatives to A. Therefore, a business model needs to provide assumptions on what economic profits the proposed values might generate, and what costs their provision might cause. The second part clearly demands for cost accounting mechanisms to be in place. Traditional cost accounting systems, however, are designed for production-oriented processes. In the context of EMANICS, with its focus on electronic service provisioning, Activity-based Costing (ABC) [29], [21] appears to be a better-suited method to determine accurate cost elements on a per-service basis. This is mainly reasoned by the assumption that services are produced once, while they are offered in the same or similar way many times. In this case, service production costs become less important with an increasing number of service requests/deliveries.
- **Value Network:** Similar to considerations with respect to value chain steps, business models usually cover information on how an organization is embedded into a value network. Value networks feature a wider angle across administrative borders. This includes the respective perspectives of suppliers, customers, competitors, complementors, and authorities. Interactions between actors and overall market conditions, thus, gain center stage. While the value chain determines with what steps a given organization creates value, the value network reflects with, against, or for what other players value is created.
- **Competitive Strategy:** The presence of economic profits sooner or later attracts competitors. In fully competitive markets, sustainable profits are provided only if an organization is able to gain a competitive advantage over its competitors. Therefore, business models rely also on a competitive analysis. According to Porter’s five forces model [43], this consists in investigating the bargaining power of suppliers and customers, the threat of new market entrants or substitute services, and market-internal rivalry. Based on the competitive analysis’ results, a suitable competitive strategy is chosen. Two basic alternatives exist: cost leadership or differentiation strategy. While the first focuses on cost efficiency (“offer the same services as your competitors, but offer them at a lower price”), the latter envisages to offer greater value to customers than competitors are able to (“offer better services than your competitors, and possibly charge more”).

From those business model elements presented, the EMANICS business modeling approach is developed and implemented as shown in Figure 1. EMANICS business models, thus, are determined by three models: a role model, a service model, and a charging model. Business models are put into operation by means of deployment models. Assumptions on costs and revenue potentials base on expectations gained from market forecasting. The actual incidence of expected numerical values, thus, may vary from profits possible and costs incurred. This aspect is reflected explicitly within the scope of deployment models by means of considerations on risk assessment.

Those models may be applied for any type of ICT systems, many examples presented below, however, are addressed toward the provisioning of electronic services, thus, forming the approach with a high potential for the integration of economic and technical mechanisms for the management of networks and their services.

3.2 Role Model

This section proposes a model for service promises, rather than network services. Therefore, a refinement of a role as an entity within a system — probably a network, a domain, or an IT infrastructure in general — is based on its intended function, not on whether it fulfils its commitments. The roles presented are determined as principal actors needed for a generic, multi-domain, multi-service IP QoS delivery and their interactions. These roles are largely in-line with the roles specified in the context of the EU MESCAL (Management of End-to-end Quality of Service Across the Internet at Large) project [34], [25].

In general, a role should not be understood as being identified with a particular body or entity amongst a number of collaborating entities. Each entity in a system could play several roles, pertaining to different aspects of the system. Thus, roles are identified as being related to specific promises made by the parties. Additionally, the following roles (cf. Section 3.2.3.1 and Section 3.2.3.2 with a number of associated promises) have been identified as important in the provision of broadband services.

3.2.1 The Concept of a Service Promise

Services are about committing to behavior that is valuable to a client. The approach can use promise theory [1], [2] to define this in terms of interactions between a number of agents. In this case, agents are initially peers that have no pre-decided roles. It is useful to begin with general definitions that can be specialized to the view of network services later.

Definition 1: A **Promise** is defined as an autonomous specification of future behavior by an agent.

Definition 2: A respective **Commitment** is defined as a promise that costs an agent work or effort to fulfill.

A concrete promise from one agent to another cam be expressed as follows:

$$A_1 \xrightarrow{\pi} A_2$$

where π is called the promise body, *e.g.*,

- $\pi = P$ — is a promise to another agent A_2 to do *P*.
- $\pi = U(P)$ — is a promise to use/accept or receive the offer of *P* promised by another agent.

A service is not necessarily something that is provided (given). It can be the opposite — something taken, e.g., a backup service takes data from a client. Thus, the direction of the arrow does not reflect a flow of data. A promise body is a tuple, in general, which contains a type, *e.g.*, a promise of web service or a promise to make a backup, and a constraint, *e.g.*, a web service in less than 5 ms or backup every hour on the hour.

The agents can already be classified into some prototype roles, and this view can be extended below.

Definition 3: A **Service Provider**, as outlined above in Section 3.2.3.1 is any agent S that makes a commitment to one or more external agents.

In promise theory, a service is associated with a typed promise made by a service provider.

Definition 4: A **Client** is any agent who enters an agreement that includes a promise to use/accept a service.

Definition 5: An **Agreement** is a sustainable bundle of bilateral (two-way) promises between two agents. Both agents make promises to one another. These might include promises of payment.

Note that a collection of promises is not necessarily consistent or sustainable. An agreement has to be worked out so that it is of mutual benefit to both parties, from each of their perspectives. Thus, a service agreement is an agreement between a service provider and a client and a Service Level Agreement (cf. Section 4.1.1) is defined as a subset of a service agreement, which concerns the scope of the commitment made by the service provider and the scope of the promise to use the service by the client.

3.2.2 Concept of a Role in Promise Theory

Promises provide a notion of a relationship between autonomous parties in a system. Roles are associated with types of promises, i.e. categories of promise bodies π. In this respect, a collection of agents can be associated with a role, given that the members of the collection make (or receive) the same type of promise to (from) some third party. In other words, we are guided by the typed structure of the graph alone.

For example, web servers or clients, are roles. Roles are effective labels for groups of nodes. In this respect, a group is defined as follows. The notation for a promise from one set of nodes to another set of nodes can be defined as follows:

$$\{n_i\} \overset{\pi}{\rightarrow} \{n_j\}$$

Definition 6: Let *A* be a set of autonomous agents. The family of sender/receiver subsets $S, R \subseteq A$ of in a promise graph, which take part in promises with bodies Π, define **Π-Roles**. Any such subset *S* or *R* in the graph is said to play a role of (sender/receiver) type Π in the promise graph.

The set of all agent nodes $W \subseteq A$ that promises to provide web service to any agent:

$$W \overset{\text{Web}}{\rightarrow}$$

plays a role, which is called “web server”. This is called a role by association, since the members of *W* are not connected in any other way than they make the same promise.

Any combination of promise bodies can be concatenated to define compound roles or role hierarchies.

3.2.3 Example Roles

Based on these considerations a set of typical roles — mainly found in the networking domain — are introduced.

3.2.3.1 Service Providers

In general, Service Providers offer services to Service Customers (cf Section 3.2.3.2). Based on the type of service provided, Service Providers can be further divided into three categories

IP Connectivity Providers (ICP)

IP Connectivity Providers only offer plain IP connectivity services, i.e. a functionality that provides reachability between hosts in the IP address space. In order to enable this functionality, ICPs must own and administer an IP network infrastructure. As a result, Service Customers connected to the same ICP are able to communicate with each other directly, since they belong to the same autonomous IP network infrastructure. In order to enable the communication between Service Customers that are connected to different ICPs, individual ICPs need to establish provider level service level agreements with each other for reachability expansion.

Application-level Service Providers (ASP)

Compared to the plain IP connectivity services offered by ICPs, ASPs offer higher-level services, such as IP telephony and content streaming services. As opposed to ICPs, ASPs need not necessarily own and administer an IP network infrastructure; they need to administer the necessary infrastructure required by the provisioning of the offered services, *e.g.*, VoIP gateways, IP video-servers, content distribution servers. As such, for fulfilling the connectivity aspects of their services, ASPs may rely on the connectivity services offered by ICPs.

Physical Connectivity Providers (PCP)

PCPs offer physical (up to the link layer) connectivity services between protocol-compatible equipment in determined locations. It should be noted that the connectivity services may also be offered in higher layers (for the network layer, layer-3, *e.g.*, IP), however these services are mainly between specific points as apposed to the IP connectivity services offered by ICPs, which may be between any points in the IP address space. PCPs are distinguished into two main categories according to their target market: Facilities Providers and Access Providers.

Facilities providers offer link layer connectivity to other higher-level providers such as ICPs for these to interconnect their own infrastructure and for interconnecting with their peers. Facilities providers may be further differentiated according to the technology they rely upon (e.g. optical fiber, satellite, antennas), deployment means (terrestrial, submarine, aerial) and their size in terms of geographical span and customer base.

Access providers provide the connection of the end-customers premises equipment to the ICPs or ASPs equipment. They own and administer appropriate infrastructure, *e.g.*, cables, concentrators. They may be differentiated according to the technology they employ, *e.g.*, POTS (Plain Old Telephony System), FR (Frame Relay), ISDN (Integrated Services Digital Network), xDSL (Type x Digital Subscriber Line), WLAN (Wireless Local Area Network), or Ethernet, as well as their deployment means and their size in terms of covered geographical area and customer base.

3.2.3.2 Customers and Users

Definition 7: A **Service Customer** (subscriber) defines a role, which has the legal ability to subscribe to the services offered by ICPs or ASPs.

They interact with ICP/ASPs following a customer-provider paradigm, with the purpose to subscribe services to meet their communication needs and requirements. Nowadays, services are offered on the basis of respective agreements, the so-called Service Level Agreements (SLAs), setting the terms and conditions on behalf of ICP/ASPs and Service Customers for providing and requesting/accessing the services, respectively. The Service Customers can be either end-customers or even whole ICP/ASPs who act themselves as customers of other ICP/ASPs in order, *e.g.*, to extend the geographical scope of the services they provide to their own customers.

Definition 8: A **Service User** is an entity (human being or a process from a general perspective), which has been named by a Service Customer and appropriately identified by an ICP/ASP for actually requesting/accessing and using the services subscribed by the Service Customer.

The use of the services should be in-line with the terms and conditions agreed in the SLA between the Service Customer and the ICP/ASP. In essence, Service Users are the end-users of the services and they can only exist in association with a Service Customer. A Service User may be associated with more than one Service Customers using services according to the agreed SLAs of the respective Service Customers. For instance, an employee may be acting as a Service User of the services that its company, Service Customer, has subscribed to, as well as a Service User of its own subscription as a residential Service Customer.

3.3 Common Service Model

The role model presented above — as a part of the EMANICS business model — leads to the generic service model presented in this section, in which those defined roles are considered as entities within the context of service provisioning. Furthermore, most of the terms in use when describing services and service provisioning cannot be defined properly by discrete consideration, but only in close correlation.

The idea behind the following service model is to outline generic characteristics of service provisioning, independent of a concrete scenario or a specific service functionality, and even applicable to any kind of service provider (ICPs, ASPs or PCPs — cf. Section 3.2.3.1). Accordingly, a concrete service provisioning scenario can be regarded as one instantiation of this generic model.

A view on services consists of two components: the service life cycle which displays the dynamic behavior of a service and the service model which describes the composition of basically entities and interactions inside a service.

3.3.1 The Service Life Cycle

Starting with the service life cycle model, a division of the service life cycle into 7 phases is proposed, which are a refinement of [24].

In the offering phase (1) the service provider makes an offer for service provisioning to the customer. In the negotiation phase (2) a contract between provider and customer is settled. This contract contains a detailed description of the service functionality, the service level, adoptable rates, and contract penalties. During the implementation phase (3) the service functionality is implemented by the provider. Implementation and testing (4) means the (physical) establishment of the service within a productive environment. After acceptance by the customer, the service is put into operation (5). From here, the service can switch into the change phase (6); and back again into operation. Upon termination of the service, it runs through the decomposition phase (7).

3.3.1.1 Common Service Model Details

The common generic service model, developed in [17] and refined in [18], aims at two main targets: First, popular and frequently used terms in the scope of service orientation will be defined by a structured analysis of entities, roles and interactions within the service life cycle. And secondly, the generic service model shall be applicable to concrete service provisioning scenarios.

Figure 2 shows the service model in a graphical overview, whose elements will be explained in the following and which major terms are defined.

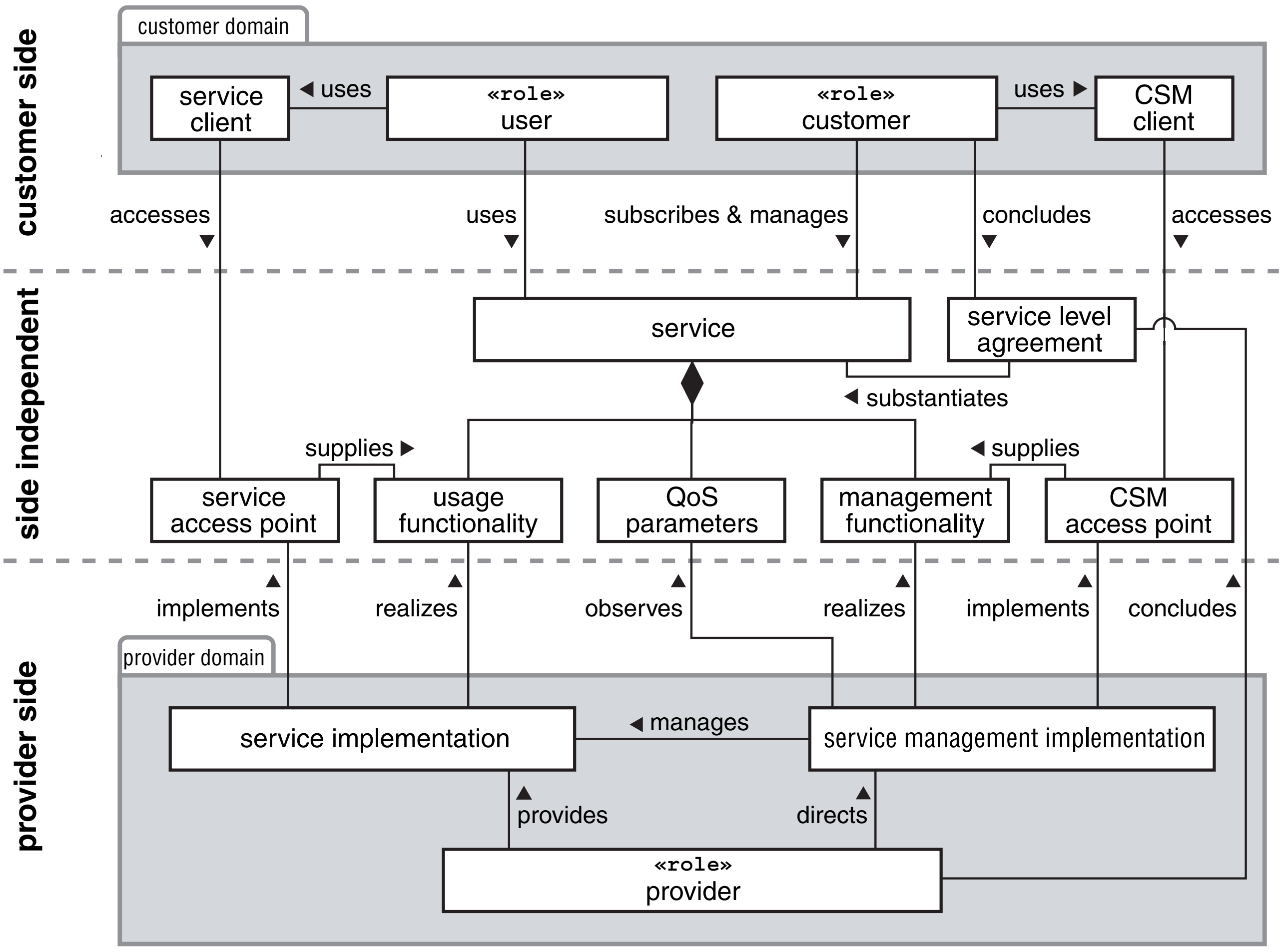

Figure 2: Service Model — Service View

Definition 9: A **Service** describes a functionality, which a service provider offers to a customer domain at a specified interface, assuring a certain service quality. The offered functionality implies usage as well as management functionality.

Components in the common service model cover a wider range. Thus, the following list if specifications cover the key terms applied within the service model. Even though a single term determines a well-specified definition, only the generic term is instantiated as a definition.

Definition 10: The **Functionality** of a service is understood as a set of interactions taking place between the provider domain and the customer domain. We differentiate between usage functionality and management functionality.

- **Usage Functionality:** The usage functionality contains all interactions between the provider and the customer domain which represent the essential purpose of the service.
- **Management Functionality:** Every interaction taking place between the provider and the customer domain which is required to operate the service, but cannot be assigned to the essential purpose of the service, belong to the management functionality of this service. This also comprises all charging activities as they are addressed in the charging model for services in Section 3.4.

Definition 11: The quality for any functionality, and thus both kinds of service functionality is specified by **QoS Parameters**. Usually, when the contract is concluded, bounds for a tolerable QoS are specified by determination of concrete values.

Definition 12: A **Service Interface** includes all entities, which are required by the user in order to provide the usage functionality. Two kinds of interfaces appear in the Service Model: the service access point and the Customer Service Management (CSM) interface.

- **Service Access Point (SAP):** The service access point provides the usage functionality of the service to the user as contracted.
- **Customer Service Management (CSM) Interface:** The CSM interface provides the management functionality of the service to the customer. A concrete implementation of the CSM interface may be implemented by a programming interface as well as by e-mail or telephone, by which management interactions between customer and service provider are being exchanged.

Definition 13: The **Service Level Agreement (SLA)** contains a description of the service functionality as well as definitions of related QoS parameters. Additionally, interfaces for accessing the service functionality by the customer domain are specified. The SLA is settled within the scope of a legal contract. A detailed discussion on the SLA specification and provisioning follows in Section 4.1.1.

Definition 14: Within this model **Domains** are used in order to define and delineate clear scopes of responsibility.

- **Provider Domain:** The provider domain comprises of all entities vital for providing the service functionality as specified in the SLA. The service provider is responsible for service provisioning and operates a service implementation and a service management.
- **Customer Domain:** The customer domain contains the user as well as the customer role according to the role model.

Definition 15: For deploying the service functionality, a **Client** is or multiple clients are required for accessing the SAP as well as the CSM access points. The customer domain is in charge of the proper operation of the service client and the CSM client.

Definition 16: The **Service Implementation** implements the usage functionality of the service provided.

Definition 17: The **Service Management Implementation** covers all tasks and resources, which are required to ensure a proper planning, installation, and operation of the service in line with the service agreement.

3.3.1.2 Service Aggregation and Service Hierarchies

If a service is part of the implementation of the functionality of another service, it can be seen - from the perspective of the constitutive service - as a subservice. Figure 3 shows the implementation view on the provider domain of our service model and points out, how service chains are reflected in this model.

3.3.1.3 Resources

The generic service model, introduced above, abstracts from resources within the underlying IT environment. To complete this contribution by a resource-orientated perspective, EMANICS viewpoints concerning resources: are stated as follows:

Definition 18: A **Resource** is a physical or logical item, on which the usage or management functionality implementation of a service is at least partially based upon.

- **Hardware resource (device):** A hardware resource is a physical item, which is part of a service implementation or a service management implementation.
- **Software resource (application):** A software resource is the installation of a software distribution within a productive IT environment, which is part of a service implementation or a service management implementation.

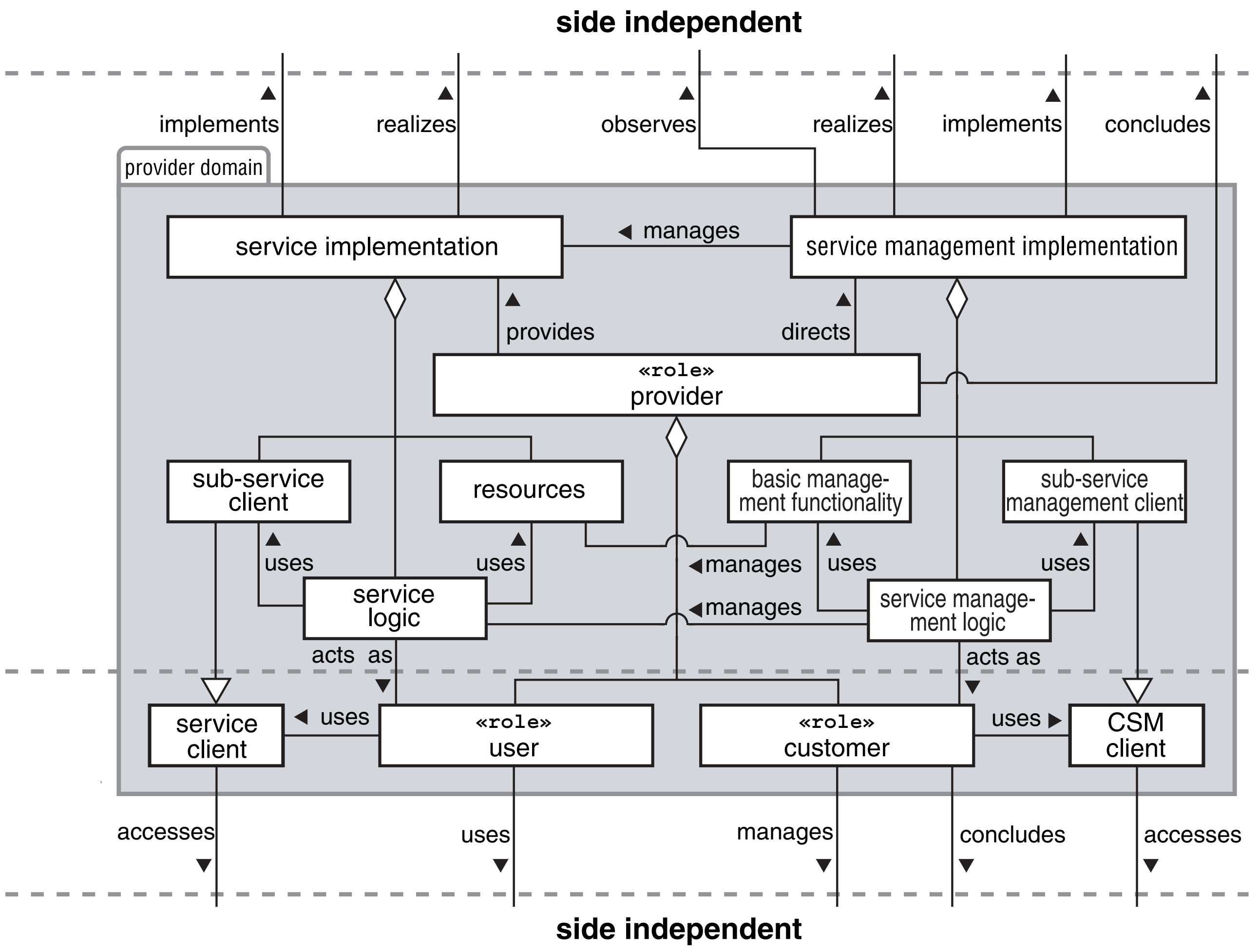

Figure 3: Service Model — Implementation View

3.4 *Charging Model for Services*

Charging determines the activity for that any type of service usage will be paid for by a customer to a service provider. Service types cover the full range from pure Internet Protocol (IP) access to Value-added Services (VAS). Thus, a number of prerequisites for an appropriate charging approach are required to be determined and defined, which are addressed below [50].

3.4.1 Prerequisites and Functionality

First of all, **metering** determines the particular usage of resources within end-systems (hosts) or intermediate systems (routers) on a technical level, including Quality-of-Service (QoS), management and networking parameters. Since the volume of data generated by metering components in various location within a network and at its access points as well as within VAS provider domains, **mediation** is intended to filter, aggregate, and correlate raw technical data, which in most cases has been collected by metering. Mediation transforms these data into a form which can be used for storing and further processing.

The next step on processing mediated data will be **accounting**. Accounting defines all summarized information (typically formalized in terms of Accounting Records, AR) in relation to a customer's service utilization. ARs are expressed in some form of quantifiable metered resource consumption, *e.g.*, for applications, calls, or any type of connections. Basically, these three functions — metering, mediation, and accounting — finalize the pure technology-driven set of parameters.

The second set of parameters are defined on the economic layer of the problem and include the definition of prices for resources, especially, unit resources, which can be metered. Thus, the activity of **pricing** — called pricing schemes as well — covers the specification and setting of prices for goods, specifically units of networking resources and units of services in a market situation, which may differ according to country-specific deregulation aspects, competitive situations, or other influencing aspects. This process may combine some technical considerations, *e.g.*, resource consumption, and economical ones, *e.g.*, applying tariffing theory or marketing methods. Prices may be, a.o., calculated on a cost/profit base or on the current market situation. To specialize the detailed algorithms for those prices, **tariffs** define those ones used to determine a charge — thus, the mapping of technical parameters into economic units, typically, monetary units — for a service usage. They are applied in the charge calculation activity for a given customer to the service he utilizes. For calculating charges the tariff may contain, *e.g.*, discount strategies, rebate schemes, or marketing information.

Therefore, **charge calculation** covers the complete calculation of a price for a given AR and its consolidation into a Charging Record (CR), while mapping technical values into monetary units. Therefore, charge calculation applies a given tariff to the data accounted for and stored in the accounting data base. To be able to identify the responsible customer for a set of charging records, the **identification** allows for the mapping of data within the accounting data base onto customer IDs.

Finally, the **billing** defines the collection of those CR, summarizing the charging content, and delivering a Billing Record (BR) to be included into an invoice/bill — including an optional list of detailed charges accumulated over a billing period, typically a month, a week, or even a day — to a user. The **payment** of this bill may be undertaken in a pre-paid or post-paid manner, each of which does pose certain requirements onto the underlying management system as well as network infrastructure. While pre-paid schemes may use pre-paid service cards or credit cards, post-paid schemes may be supported by electronic funds transfers or credit card payments.

Finally, the term **charging** is used within this work as an overall term, depicting all tasks required to calculate the finalized content of a single or multiple BR. Sometimes in the literature, the term billing is utilized instead, however, it includes the full handling of invoices and customer relation management tasks, which are of clearly second priority of EMANICS, since the technological and economic aspects are addressed in an integrated manner.

3.4.2 Comparison of Different Charging Systems and Approaches

Note that the terminology used in different standardization and working domains — all of which are addressing communications as such, though — varies largely. Therefore, the following paragraph summarizes the key areas besides the IETF's (Internet Engineering

Task Force), mainly including the wireless communications domain. As discussed in [30], today's 3G systems in Europe are based on UMTS (Universal Mobile Telecommunication System) Release 99, while its counterpart consists of CDMA2000 (Code Division Multiple Access) 1x Release 0 for circuit-switched services, together with CDMA2000 EV-DO (Evolution-Data Optimized) Release 0 for packet-switched services. Most of the terminology used in these specifications for different 3G (3rd Generation) releases does not match one-to-one with the IP world. Thus, those correlations are shown in Tab. 1. The right column is based on 3GPP's (3rd Generation Partnership Project) vocabulary definitions [15], which are also used in 3GPP2's online charging specifications.

Table 1: Comparison of Charging Terminology — Based on [30] and Extended

All IP-based Networks/EMANICS	**3G Mobile Networks**
Metering	Collecting charging information
Accounting	Charging
Accounting Record (AR)	Charging Data Record (CDR)[a]
Charging options	Billing arrangements, Payment methods
Pre-paid/post-paid charging	Pre-paid/post-paid billing
Charge Calculation	Charging mechanism
Charging Record (CR)	Transferred Account Procedure (TAP)[b]
Billing and parts of charging	Rating (parts of rating)
Billing Record (BR)	Transferred Account Procedure (TAP)[c]
Inter-/Multi-Domain Charging/Billing	Accounting

a. The details and content of such CDRs depend on the service offered, particularly being dependent on the packet-switched service or the circuit-switched service. Even different names may be used, *e.g.*, M-CDR for a CDR of a mobile service.

b. A new set of records was developed in terms of TAP2 and TAP2+, which support the process of operators to bill for new services and to provide the additional information required [22].

c. Partly the use of TAPs includes the input data for billing systems, however, TAPs cover the combination of All IP-based terms Charging Record and Billing Record, mainly addressing wireless services and roaming support.

3.4.3 Extended Charging Model

Thus, the model of charging support in any networking domain is defined as shown in Figure 4, partially extended from [49].

This model consists of those 8 types of components, which address on the bottom the three types of resources to be charged for: computational resources — those originating from any type of traditional or advanced grids as well as from single computers —, networking resources — at this stage limited to IP-based parameters such as bandwidth, error rates, or delay —, and services, which do exist as resources or value-added services going beyond the pure IP access.

At the second level, all of those terms defined above do interact as shown. While the mediation takes metered data and aggregates them as defined, the accounting component covers the first storage capability of that type of data flow: the accounting data base stores Accounting Records (AR) — either locally or in remote locations, mainly depending on the scenario. While prices and tariffs are stored in respective data bases, the IDs of customers, users, and their grouping and stored in additional customer/user data bases.

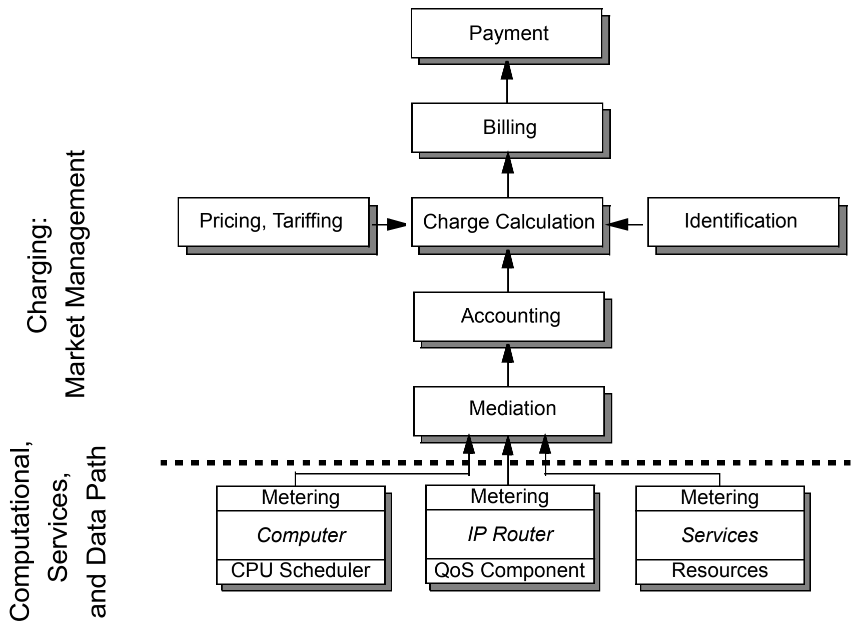

Figure 4: Charging Model — Extended View based on [49]

Additionally, the charge calculation stores, if required, Charging Records (CR) for each and every single transaction performed, when AR are taken out of the accounting data base. These CRs are handed to the billing component, which generates bills or invoices, which in turn are stored in Billing Records to be handed over to the respective payment implementation of the payment module. Payment itself is outside the scope of EMANICS, but it may be operated in the traditional cash-based payment, via an electronic funds transfer, or a credit card payment. Those types of payments are considered as off-line payments and may be complemented by on-line payment methods, which integrate — at least to a certain extend — the accounting, charge calculation, and billing function —, to ensure that any services being used a any point in time will be paid for right away, even during their use.

3.4.4 Brief Overview of Internet Pricing Schemes

The numerous Internet pricing schemes proposed over the last years (for a closer overview see, *e.g.*, [51]) cover a range from highly dynamic and market-oriented to simple and inflexible approaches. The major milestones proposed so far look as follows [44]:

- **Auctions** provide a very accurate and highly flexible feedback mechanism about the market situation. Unfortunately, implementing auctions on packets would yield a size of expenses which today is considered to be unbearable.
- **Volume-based** approaches reduce the accounting task to the volume as an important parameter in terms of resource usage, but still require a rather accurate monitoring of the amount of data travelling through the network.
- **Classification** is another well-established concept for reducing complexity, *e.g.*, by defining service profiles for individual users and tags packet demand to be in- or outside this profile. Similar ideas have led to classifying offered services themselves in terms of QoS, thus forming one of the core points of the Differentiated Services framework. Here, as a rule of thumb, higher QoS yields higher prices.

- **Edge Pricing** deals with another aspect of complexity reduction, now in terms of locally concentrating the distributed nature of pricing decisions by shifting them to the edge of the ISP. This concept is charming for its simplicity, decoupling complex price negotiations between customers and various ISPs into a series of bilateral ones, as well as for its transparence towards the user.
- **Flat Rate** approaches are finally the only ones that are universally accepted to be technically feasible in today's Internet. They consist of a monthly charge which includes unlimited usage of the Internet services.
- **Cumulus Pricing Scheme (CPS)** proposes a variable rate scheme, which varies over long time-scales only. To bring market forces into play it provides a feedback mechanism, where this feedback is not an immediate one, but requires the accumulation of discrete "flags" according to user behavior.

While those milestones look as four fixed categories, they do not form a commonly agreed upon classification so far. The major reason for this is based on the fact that many practical pricing schemes for Internet services are constructed out of many different components and those pricing categories are combined.

Furthermore, the problem occurring is growing worse, since marketing terms and company-specific names are utilized to defined "new" schemes, which in fact show a slightly changed or even different set of parameters being applied, but the core principle hidden behind these names remains similar. Thus, this overview briefly outlines an overview and does not determine a full-fledged and justified classification.

3.4.4.1 General Charging Requirements

A closer look into the different charging approaches reveals that any proposal for tariffing Internet services has to cope with several general requirements [44], which show three main Requirement Types (RT):

- **Requirement Type 1: Customer.** Over the last years, there has been an extensive discussion on preferences customers show towards dynamic tariff schemes. *E.g.*, within the INDEX [14], CATI [50], and M3I [35] projects it has turned out that especially transparency and predictability of charges are of major importance to the Internet customer.
- **Requirement Type 2: ISP — Economic.** Economically, the ISP is interested in running the network efficiently, *e.g.*, by maximizing network utilization or total revenue. Thus, pricing schemes represent an important interface to the customer, as prices may be used to indicate well-behavior or misbehavior of the customer or to signal the congestion state of the network, i.e. allowing the ISP to communicate the relationship between current customer behavior and overall system status.
- **Requirement Type 3: ISP — Technical.** Each Internet pricing scheme depends heavily on the existence of tools for technical accounting. There is a vast field of possibilities as to which detail data about the network status are to be obtained since technical conditions may vary enormously.

Thus, a charging approach to be applicable, acceptable, and economically viable needs to optimize those three requirements types for the given situation.

3.4.4.2 Time Scales

In consequence, the following four different time scales have been identified as being highly relevant for Internet charging schemes [44]. Those time scales are becoming more and more important since the efficiency to be achieved — either for the technology deployed and the charging scheme established — needs to be optimized continuously for any provider and any arbitrary set of services to comply (a) with the business model's goals and (b) with the competitive strategy chosen.

- **Atomic (communication-relevant):** This involves sending packets, round-trip times, and managing feedback between sender(s) and receiver(s).
- **Short-term (application-relevant):** This-time scale is concerned with the usual duration of applications like file-transfer, video-conferencing, or IP phone calls. The tasks of accounting and metering are closely related to these activities.
- **Medium-term (billing-oriented):** The time-scale for performing billing actions depends strongly on the usual human lifestyle habits of humans, *e.g.*, monthly payments of rents, phone charges, or newspaper bills
- **Long-term (contract-specific):** The largest time-scale in this context is the duration of contracts between customers and ISPs, which usually varies from several months to years. Note that contracts between ISPs may be shorter.

Figure 5 sketches the relationship between these time-scales, the respective management tasks, and communication contents as well as various pricing schemes. Note that the technical support for a particular pricing approach needs to be technically efficient to ensure that a potential economic advantage can be kept in a productive and operational system.

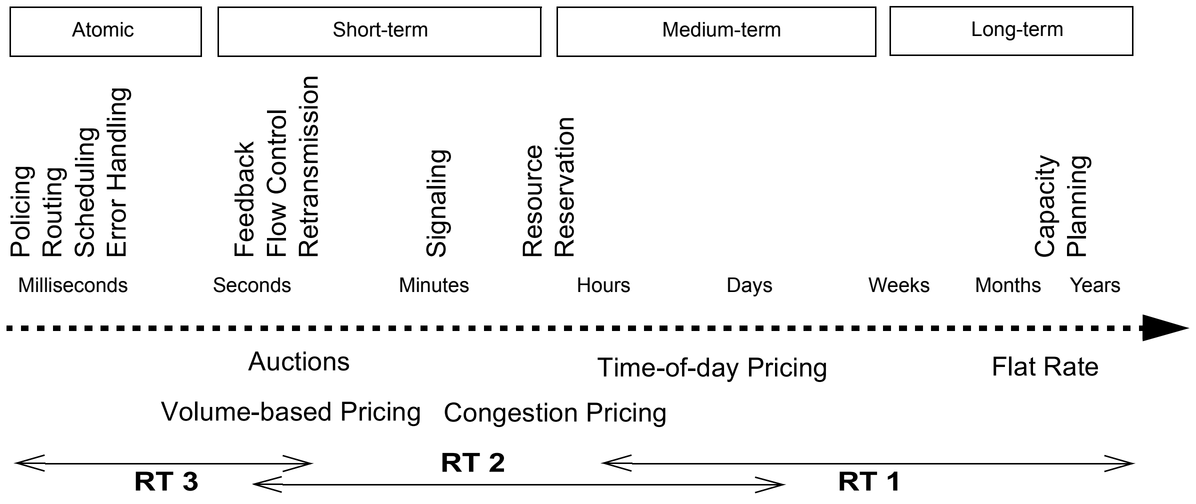

Figure 5: Charging Time Scales

3.5 Deployment Model

Business models are instantiated into operation by means of deployment models. Thus, different application domains or application areas will show different deployment models. To outline briefly the key steps to be undertaken to "instantiate" a business model for an operational case, the following example of a telecommunications company has been taken.

Telecommunications company T, defines the business model being of interest to them by determining the value proposition, addressing the market segment, identifying the value chain, summarizing the cost structure and the potential profit, outlining the value network, and investigations on the competitive strategy.

Those very minimalistic steps to be undertaken are exemplified in a small example as follows:

- **Sample Value Proposition:** The objective for T is to create value by offering a set of voice services, Voice-Over-IP (VoIP) services, and a number of Value-added Services (VAS). This will be based on a competitive assumption, since other telecommunication companies will be in the market. Thus, those services are perceived as the relevant form of electronic products, which provide a certain amount of utility to the service usage for which, in turn, the service customer will be charged.
  Thus, the roles of the provider (T) and the number of customers (C) — originating from the role model —, if any mentioned explicitly, are determined. In addition, the services have been “instantiated”, explicitly naming the relationship between the C and those services S — originating from the service model.
- **Sample Market Segment:** To satisfy customer needs in country C and addressing its specific market environments, both, service providers and service users do have different incentive settings, which determine behavior patterns. Accordingly, the marketing activity in this sample addresses the B2C area. Therefore, the SLA is based on an ISP to user contract and it contains SLS data for traditional voice services, the VoIP services, and all VAS with a certain guarantee level.
  Thus, the different customers C may be subdivided into different users (U) and their groups, covering their different type of service usage behavior, which forms a refinement of the role model in accordance to the specific market segment.
- **Sample Value Chain:** In a very simplistic case the value chain consists of provisioning activities of T, the monitoring activities of T to ensure that quality metrics are met and kept, and to provide for a charging system, in which all three types of services can be accounted for and billed at the end of the month. The pricing for the voice service will follow a time-based and region-based approach. The VoIP traffic will be priced based on the amount of data leaving the customer (source of origin), which can be simply accounted by a traffic meter. And the set of VAS will be priced on an event basis, in which case certain actions, such as click on pre-determined buttons will start a certain application and the respective accounting operation (counting). On one hand, the competitive strategy (as exemplified below) an actor takes, will effect the value to be achieved, on the other hand, the this knowledge will form the key business advantage of T, as will be hidden from inspection by organization-external entities.
  Thus, the charging model has been “instantiated” for the set of services C, even probably based on a per customer contract, if required for personalization reasons. In the latter case a single service S instantiation may show a different charging scheme to be applied.
- **Sample Cost Structure and Profit Potential:** In the example shown, an investigation may have shown that the offer of VoIP services will increase the customer basis of T, which will lead to an increased usage of those VAS offered. Thus, the economic benefit of T is given by (a) a larger customer base and (b) a higher utilization of services. A detailed set of assumptions will be required to ensure that those conclusions can be justified. Furthermore, the set of cost accounting systems has to be determined, to make sure that electronic service provisioning characteristics are met. Thus, means to obtain accurate cost elements on a per-service basis are essential. A sample for a slightly different domain and the application of Activity-based Costing is discussed in Section 10.4.
- **Sample Value Network:** Here as well, similar to considerations with respect to those value chain steps, the deployment models covers the information on how T is embedded into the overall value network of the telecommunications industry. Since the multi-domain view does form the major influencing factor here, technology, regulatory,

and customer demands have to be considered. Of course, such a broad range of factors can not be simply outlined for the example taken, however, besides telecommunication customers, other telecommunication competitors and complementors as well as governmental authorities will determine under which conditions the set of services will be beneficial for T.

- **Sample Competitive Strategy:** As soon as economic profits appears for T, competitors are attracted. To ensure that this advantage remains in place, the competitive analysis will form the basic instrument to determine new sectors, new segments, new technology, new services, under which the current position can be strengthened or extended.

4 Existing Key Mechanisms

Mechanisms in support of multiple network management tasks in a distributed system consist out of two different types:

- Technical mechanisms and
- Economic mechanisms.

While the former ones are quite well-known in traditional management schemes, such as Service Level Agreements (SLA) and policy-based management approaches, the latter ones determine a new trend in management, driven by various needs to be discussed in the following, where economic measures, payments, and business model-driven influences enhance the traditional view of network management tasks into the economic management of networks. Both of those layers are seen from an integrative perspective as indicated in Figure 6, which is supported by a vertical integration module.

Integration in that sense determines that the set of parameters to be used, *e.g.*, for a QoS Management mechanism (residing in the lower layer) need to be reflected in the SLS (residing in the upper layer), which in turn is part of the SLA. Additionally, this SLA influences the cost model to be utilized for a given provider and technology in use, which finally determines in one way or another the pricing mechanism chosen, of course, addressing the deployment model assumption as indicated in earlier sections.

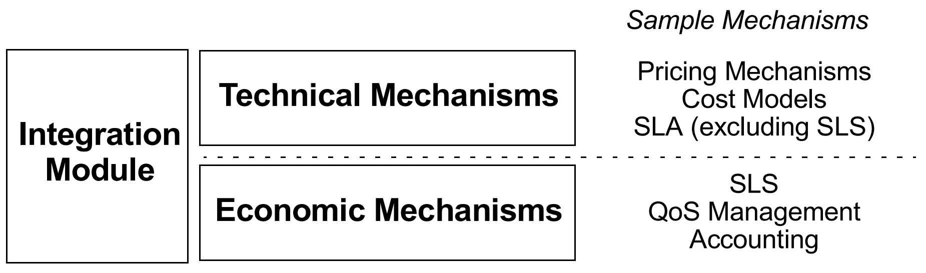

Figure 6: Layering Approaches for Mechanisms

This two layered structure may not remain static in the future, since a conceptual discussion reveals that, on one hand, there seem to be good reasons for including an intermediate layer, in which the integrative view of technical and economic parameters may need to become visible. On the other hand, such an intermediate layer may also become an interface layer only, at which a predefined Application Programming Interface (API) may reside. However, as long as no implementation model is required to be defined, the finalization of this open issue can be deferred to a later stage.

Thus, this section does address in an overview type of style those two categories and it summarizes the major and highly important mechanisms known today. Furthermore, open issues and emerging problems to be solved are summarized.

4.1 Technical Mechanisms

To be able to address the technology basics for defining a service quality and delivery action, SLAs form the basics. SLAs specify services and their characteristics, which shall be followed in the service provisioning process to ensure a customer/user to provider relation based in common grounds. In addition, this initial specification has to be followed by a negotiation scheme, under which those two roles will be able to agree on parameters, parameter sets, values, and characteristics to be applied explicitly in a given service usage.

This process may be guided by policies, determining rules, which do address context-specific details to be applied for a certain case. Finally, the accounting in the technical sense addresses the collection — and possibly aggregation — of data, which originate from measurement points and define the service-specific definitions of resources, actions, and events, which have taken place in the service usage — and possibly the set-up and tear-down — phase.

4.1.1 Definition of an SLA Specification and its Provisioning

There are several definitions and terms of a what a SLA is respectively should be. In the following some of the definitions are reviewed. If an agreement is considered to be a a bilateral promise between a service provider and a customer (and vice versa), then an agreement constrains the behavior of both parties, otherwise it is merely a promise.

Definition 19: A SLA is defined as bilateral bundle of promises between a customer and a service provider, in relation to the provision of a service. Another definition is that a SLA specifies the service levels (quality, security) that a provider needs to offer to a customer at a service access points at a certain cost [12].

Sometimes, also the term Operation Level Agreements (OLA) appears. It refers to agreements about the provided quality internally within a provider. They include no legal aspects but just technical parameters and thresholds (i.e., SLS). In case of provider chains, the term Underpinning Contract (UN) comes up to identify an agreement about service levels between a provider and its sub-providers.

There is no common definition of what an SLA entails. There seems to be a consensus that an SLA must talk about performance characteristics that are measurable, hence verifiable. Other legalities might be discussed. These might include reimbursement for loss, breach of copyright, or privacy due to incompetence. However, the price of a service unit is an important part of the agreement.

SLAs require a clear unambiguous language, since they are legal documents. Modal logic has played a role in developing such language syntax.

The vision is to formalize a SLA in order to automate service provisioning as far as possible. The vision of a service provider and also strategic competitive advantage is flow-through provisioning, which means that a customer reviews a set of services offered to him by a provider, selects an appropriate service, chooses one of the available service quality options, then simply places and accordingly formed service order, and waits for its almost instantaneous fulfillment. The service provider, who receives such an order, would have automated provisioning systems that activate, monitor and manage the ordered services with minimal additional manual input.

According to [47], the following points are warranted:

- Introduction
- Scope of Service
- Performance, Tracking and Reporting
- Problem Management
- Compensation
- Customer Duties and Responsibilities
- Warranties and Remedies
- Security
- Intellectual Property Rights and Confidential Information
- Legal Compliance and Resolution of Disputes
- Termination of Agreement
- Signatures
- Schedules

Hence, SLAs include a legal and a technical part. For an applicable SLA, the technical part consists of parameters that have to be defined explicitly. However, the semantics of a QoS parameter can be quite ambiguous. Issues appear such as how to determine the availability, *e.g.*, at the SAP, should the availability of the SAP be tested from the provider's and customer’s side, respectively, what mechanisms are used for testing such as ICMP (Internet Control Message Protocol) or SNMP (Simple Network Management Protocol). This leads to the conclusions that parameters should be described as precise as possible by identifying also basic values and metrics i.e., how to calculate the parameters.

Thus, the following list of sample parameters may serve as an insight to those:

- Availability (*e.g.*, times of day, calender days, or hours)
- Mean response time
- Mean data rate
- Time scale over which means will be computed and enforced
- Probability of achieving mean data rate
- Probability of achieving mean data response time
- Accuracy limits
- Penalties if service is unavailable
- Penalties if a user exceeds their limit
- A schedule for meetings

The representation of an SLA can take many forms. If there is a standard syntax for SLA, then its possible representation as XML (eXtended Mark-up Language) is simply a mapping. The approach shown in [32] — the Web Service Level Agreement (WSLA) language — proposes to “express the operations of monitoring and managing the service”. These tasks “may include third parties (such as Management Service Providers) that contribute to the measurement of metrics, supervision of guarantees or even the management of deviations of service guarantees”. Additionally, the work of [55] describes which “role an SLA plays in the Internet Data Center (IDC) and how it helps assure that one's reputation stays intact. It also includes sample agreements that can be used as templates".

In general, there are several approaches one might take:

- Verbal:
  Work on SLAs must somehow end up with this, since the content has to be discussed
- Logical:
  E.g., some form of temporal logic about what must happen
- Promise theory approach:
  Will become highly useful for more complicated agreement graphs

4.1.2 SLA Negotiation Mechanisms

A lot of research has been done on negotiations from various perspective. Researchers of the agent technology conducted a strong stream of work to support negotiation activities [5]. The increasing importance of Grid computing and virtual organizations has increased the importance of SLA negotiation mechanisms as well. For example, in the EU project GEMSS (Grid-enabled Medical Simulation Services) [19] the negotiation of SLAs between competing service providers is solved by agent technology. The macro QoS negotiation between a client and multiple service providers is based on the FIPA (Foundation for Intelligent Physical Agents) reverse English auction protocol [16].

The goal of the OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee [37] is to automate the negotiation process between two negotiation parties for bargaining different technical issues in the context of Collaboration Protocol Profile (CPP). As a result, an agreement between two negotiation parties is expressed in the format of Collaboration Protocol Agreement (CPA).

Traditionally, there are two types of negotiations, which can be distinguished. On one hand, distributive negotiations (also known as zero-sum and competitive negotiations) are classified as win-lose negotiations. For example, one party reaches its goals and the other party must fail to realize it. On the other hand, integrative negotiations (also known as collaborative and cooperative negotiations) are classified as win-win negotiations. The major reason to adopt an integrative negotiation is that integrative negotiation always reduce the likelihood that negotiations will fail, by making it possible to locate options that satisfy parties' ultimate expectations.

The negotiation process consists of several phases:

- Pre-negotiation for understanding the negotiation problem,
- Conduct of negotiation in terms of offers construction and counter-offer, and
- Post-settlement computation of possible offers that dominates the most recent compromise.

Certainly, the negotiation between a customer and a provider can span several iterations, and refers to the negotiation of service levels at the Service Access Point (SAP).

The negotiation differs for individual or mass services. In case of individual services, a customer asks for services and service levels that a provider tries to implement, also if he needs to change his infrastructure or change contracts with his suppliers. Certainly, such negotiations are cost-driven. In case of mass services, a customer selects a service or a service package from the available set of services with predefined service levels. The negotiation process is in such a case very limited, almost not existent.

SLA negotiation mechanisms address mainly three aspects:

- the negotiation objects
- the negotiation protocols, and
- the decision making models.

Negotiation objects refer to issues over which an agreement must be reached, the identification of attributes which are negotiable or non-negotiable, respectively, and the type of attributes, whether they refer to quality and service level or price and penalties.

Negotiation protocols identify and specify who (especially, which participants) can do what activity (*e.g.*, an event or actions) at what time. There exist various approaches to identify an adequate negotiation protocol. An approach is to use the Contract Net protocol [48], which is a high-level protocol for achieving efficient cooperation in agent-based systems. Another example is COPS-SLS [52], an extension of COPS, that allows for the configuration of domain policies regarding service levels and the automatic negotiation of service levels within the domain policies. Finally, WS-Negotiation [26] is an approach applicable to Web Service (WS) environments. WS-Negotiation is an independent XML language that can be applied to different types of agreement templates. Negotiation on business information is a much more complex subject, and WS-Negotiation is targeting on the business parameters and legal matters.

In general, decision making models specify the negotiation strategy and the reasoning model. They depend on the negotiation objects and protocols.

4.1.3 Definition of Common Policy-based Management Basis

Besides the SLAs and their negotiation, a flexible approach on decisions based on current status or conditions is required to achieve an adaptive management approach. Therefore, this section introduces essential and generally admitted concepts in policy-based Information Technology (IT) management. Additional information may be obtained at [53].

4.1.3.1 A Universal Policy Architecture

Basic components of policy-based management are [33], [18]:

- **Policy Repository:** A Policy Repository is the maintenance component for a whole set of policies. It represents a knowledge base for the IT management, in which policies are stored persistently.
- **Policy Decision Point (PDP):** The PDP is the logical place/location, where the decision whether a policy shall be executed, takes place. Therefore the condition part of the policy must be evaluated. One central PDP is part of most policy architectures.
- **Policy Enforcement Point (PEP):** The PEP is the logical place/location, where policies are being executed. The PEP has to put the action part of a policy into practice.

As depicted in Figure 7, Policy Repository, PDP and PEP in combination build a simple, universal policy architecture. For communication between PDP and PEP, the Common Open Policy Service (COPS) protocol [13] and its secure transfer COPS over TLS [54] via Transport Level Security have been recommended by the Internet Engineering Task Force (IETF).

PEPs may be distributed over the whole managed IT environment. For example, every network node (*e.g.*, routers, switches) can hold its own PEP. Then the (central) PDP has additionally to select the PEP, which has to execute the evaluated policy.

Usually, policies are event-triggered and can be executed in parallel, which means that one event may trigger more than one policy at the same time. PDPs are assumed to be without memory. The evaluation of a policy-condition relies exclusively on the boolean expressions in the condition part of the policy. Policies which have been executed before have no influence on the evaluation process.

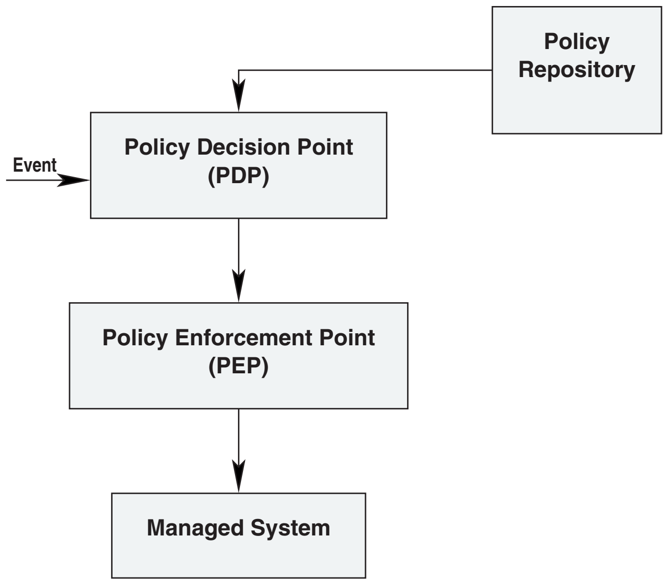

Figure 7: A Common Policy Architecture

4.1.3.2 Policy Hierarchy

An important feature of policies is the possibility to express management instructions in different levels of abstraction and detail. Hence, policies are applicable for all kinds of management disciplines, i.e. element, network, system, service, customer or business management. Policies of a lower level are derived by refinement of higher level policies. In this process the related entities and actions are rendered more precisely or narrowed down.

- **Corporate Policies:** Corporate policies (or High Level Policies) can be assigned to Business and Customer Management. They are directly derived from the business objectives and have a strategic alignment. The level of detail is low, and technical aspects are of minor prominence. Often these policies are described informally as text.
- **Functional Policies:** Functional Policies are a Refinement of Corporate Policies and assigned to distributed system and service management. They operate on management platforms and deploy the provided services, such as CORBA Services [9] or OSI Systems Management Functions [27]. Policies on this level have a formal syntax and are executable.
- **Low Level Policies:** Low Level Policies build the bottom layer as they are assigned to network and element management. It is essential that this kind of policies operates directly on managed objects or sets of managed objects.

4.1.3.3 Policy Languages

The common structure of a policy is derivable from the most important publications in the sector of policy-based management:

- **Subject:** defines who will execute the policy (often specified by a role) and in so far declares the responsibility.

- **Target:** specifies objectives which are affected by the policy.
- **Event:** specifies the event which has to occur in order to start the policy evaluation.
- **Action:** defines the action which will be performed on the specified target.
- **Condition:** defines a constraint for the execution of the policy. The condition part can be regarded as precondition. If no condition part exists, the policy is always executed when the specified event occurs.

Popular policy languages for IT management like Ponder [10] and ProPoliS [11] follow this structure.

4.1.4 Accounting

Accounting, charging, and billing have been identified as three essential processes involved in the commercial exploitation of products and services, in particular Internet and telecommunication services. Within the Internet Engineering Task Force (IETF) the Authentication, Authorization, and Accounting (AAA) working group focuses on developing requirements for authentication, authorization, and accounting as applied to network access and proposed a generic architecture [31].

In general, the provisioning of commercially exploitable services requires that users are correctly authenticated and that they dispose of sufficient authorization privileges in order to access the service. Once that access is granted, the consumption of the resources and the usage of services has to be adequately accounted for. Generally, **accounting** can be seen as the collection and aggregation of data about resource consumption and service usage. This includes the fully decentralized control of data gathering (via metering), transport, and storage of accounting data. Besides the usage-to-user mapping, the main goal of the underlying accounting system is the storage of accounting data and data management.

Furthermore, **accounting policies** determine which data is stored, how often and in which way data is collected and made available to other functions. The accounting information, usually stored in so-called accounting records, can for instance be used for capacity and trend analysis, load balancing, cost allocation, billing, or auditing. From the point of view of commercial service provisioning, accounting hence is perceived as the central component in a row of supporting processes:

Accounting relies on identification of users (authentication) and the determination of respective user profiles (authorization). Stored accounting records then feed into downstream processes, embracing well verification of service level agreement compliance (auditing), calculation of non-monetary values (charging), and finally cleared monetary values (billing), summarizing resource consumption and service usage in condensed form, thus presented in a readable and understandable way for service customers.

For systems being in place today, the generic AAA architecture [31] describes the set of general network components required and their functionality on an abstract level. Concerning this architecture, a layered approach and structure for AAA protocols has been proposed. While initially the RADIUS protocol (Remote Authentication Dial-In User Service) [46] has determined the widely deployed AAA protocol – mainly in support of authentication and providing also accounting functionality [45] –, the protocol termed Diameter [3] defines the next generation AAA protocol. It is based on RADIUS, but Diameter is much more flexible, and it provides reliable data transfer, fail-over mechanisms, better error handling, and security mechanisms. The Diameter protocol consists out of the generic base protocol and various Diameter applications. Diameter allows for the specification of Attribute Value Pairs (AVP), which define, a.o., for accounting purposes parameters to be accounted for.

With respect to accounting records in existing systems and standards, current important accounting record formats can be summarized as follows. They belong to

- the RADIUS approach [46], defined by a fixed set of AVPs,
- the Diameter approach [45], [3], defined by a flexible set of AVPs,
- IPDRs (Internet Protocol Detail Record), defining a unified XML format [28],
- Cisco's NetFlow data [7], or
- IETF's IPFIX proposal [8] for the representation of IP flows.

Finally, other protocols in support of a principle accounting may be considered, which include COPS (Common Open Policy Service), RSVP (Resource Reservation Protocol), or SNMP (Simple Network Management Protocol), however, they are not relevant in practice for a large accounting approach due to a slightly different initial focus and a performance-sensitive negative externality.

As this existing systems overview shows, accounting has been considered a necessity for commercial telecommunication providers or IT outsourcers in order to charge their resources and services respectively, however, for a number of particular application domains accounting has been rarely implemented, such as for the domain of Grid computing. The virtualization of organizations, resources and services within a Grid environment additionally leads to requirements and questions beyond the common accounting architectures, thus forming a larger task on management of distributed resources beyond networking services. Generally, the focus of existing Grid accounting systems and tools is on the accounting of physical Grid resources as for example computing elements, storage resources, software licenses, and scientific devices. An adequate accounting of virtual, potentially complex services, as for example information services or computation services which may be offered within a multi-provider scenario has not been sufficiently addressed so far.

In order to perform an accounting for Dynamic Virtual Organizations (DVO) comprising a set of virtual resources and virtual services, a generic service model for DVOs is needed. An important aspect of the service model on the one hand is to reflect the underlying virtualization concept and on the other hand to contain different views of the provisioned virtual resources and services in regard to the level of aggregation. Thus a comprehensive service model is needed which offers the flexibility to reflect the view of a single service provider, a service aggregator as well as the view of the entire Virtual Organization (VO) by using the same basic principles. Another important aspect which has to be addressed is the specification of adequate accountable units building a basic set of service primitives any service provisioned within a VO is composed of.

4.2 Economic Mechanisms

To be able to address the economic basics for defining a service quality and delivery action, economic measures, and metrics as well as cost parameters form the appropriate basics. In addition, this initial definition is followed by a detailed view onto mobile micro-payment schemes and their respective accounting systems in support of extremely low-value transactions required for various on-line services.

Finally, the analysis of service provisioning templates — under the assumptions of SLAs — and their definition addresses (1) a customer- and provider-based SLA approach, where IP Network and Service Providers offer physical connectivity and (b) a pure Service Provider-based approach under which providers agree upon an interconnection agreement.

4.2.1 Definition of Economic Measures and Cost Parameters

Any type of service offered in a networking environment — ranging from pure IP access services to Value-added Services (VAS) — will be required to be specified and described. While the technical specification may be based on SLA, as defined above in Section 4.1.1, the economic specification does include relevant tariffing and pricing information as well, as outlined in Section 3.4.4. The description additionally may contain further information, with which the use of the service, the customer's profiles, the market segment, and the target numbers of users are outlined. This type of information can be considered as business-critical, since it defines provider-internal models and assumptions under which the publicly announced service specifications have been calculated in support of a typically economically viable business model.

Thus, a clear understanding of relevant **economic measures and cost parameters** is essential to determine the key economic drivers for a finalization of any pricing information on any type of service offered.

Definition 20: Cost parameters define the different types of costs involved in any type of service provisioning.

Definition 21: Economic measures define the range of potentially quantified metrics applicable, typically technical values derived from, *e.g.*, QoS parameters or metering parameters.

Finally, those economic measures are mapped onto service charges, which are defined in terms of monetary units per service unit utilized.

Thus, the economic measures and cost parameters of interest for an IT system will be required to map between such technical — in a normal case, well-defined, quantifiable, and standardized in many instances — specifications and the provider-internal business model assumptions. Therefore, for those number of economic measures and cost parameters it will be possible to list many and most of those in an initial investigation, however, the modeling correlations between those and the service offered, the provider's networking domain, or the legal requirements valid for such an offering can not be performed in a general manner. Only the detailed provider-specific set of assumptions — including the services, the service mix, the market situation, the competition regions, the demand curves over time, and the set of user classifications — will enable for the calculation of a detailed and a meaningful result, typically including benefit/loss information, sensitivity analysis on demand variations, and technical restrictions to be considered.

To provide an overview on economic measures and cost parameters, Figure 8 outlines the **theoretical view of costs**, which exist in any IT system of relevance. Basically, costs are subdivided into transaction costs and accounting costs, which are complemented by social costs, opportunity costs and sunk costs. While the transaction costs address those ones which are related to a specific operation, such as search, information retrieval, bargaining, negotiation, policing, and enforcement, the accounting costs cover microeconomic total costs and their different types as well as a number of costs types driven by financial accounting approaches (cost, management, and internal accounting) and business administration costs.

Besides this theoretical evaluation of costs, the **technology-driven view of costs** can be determined. Thus, to provide for an initial list of cost parameters in the technical domain of network management the following ones are based on the OSI Network Management Model FCAPS (Fault, Configuration, Accounting, Performance, Security) [56]:

1 **Fault Management Costs:**
These costs occur for identifying problems and network failures or costs for problem resolution.

2 **Configuration Management Costs:**
These costs occur for Hardware and Software inventories, costs for configuration changes.

3 **Accounting Management Costs:**
These costs include the collection and storage costs of network usage data.

4 **Performance Management Costs:**
These costs are required for network-capacity planning, i.e. network availability, throughput, delay, jitter, and eliminating network bottlenecks.

5 **Security Management Costs:**
These costs exist due to the definition and execution of security policies, *e.g.*, access control policies, logging, network protection, and risk identification. They may rise dramatically if a network is under attack and may even include liability costs.

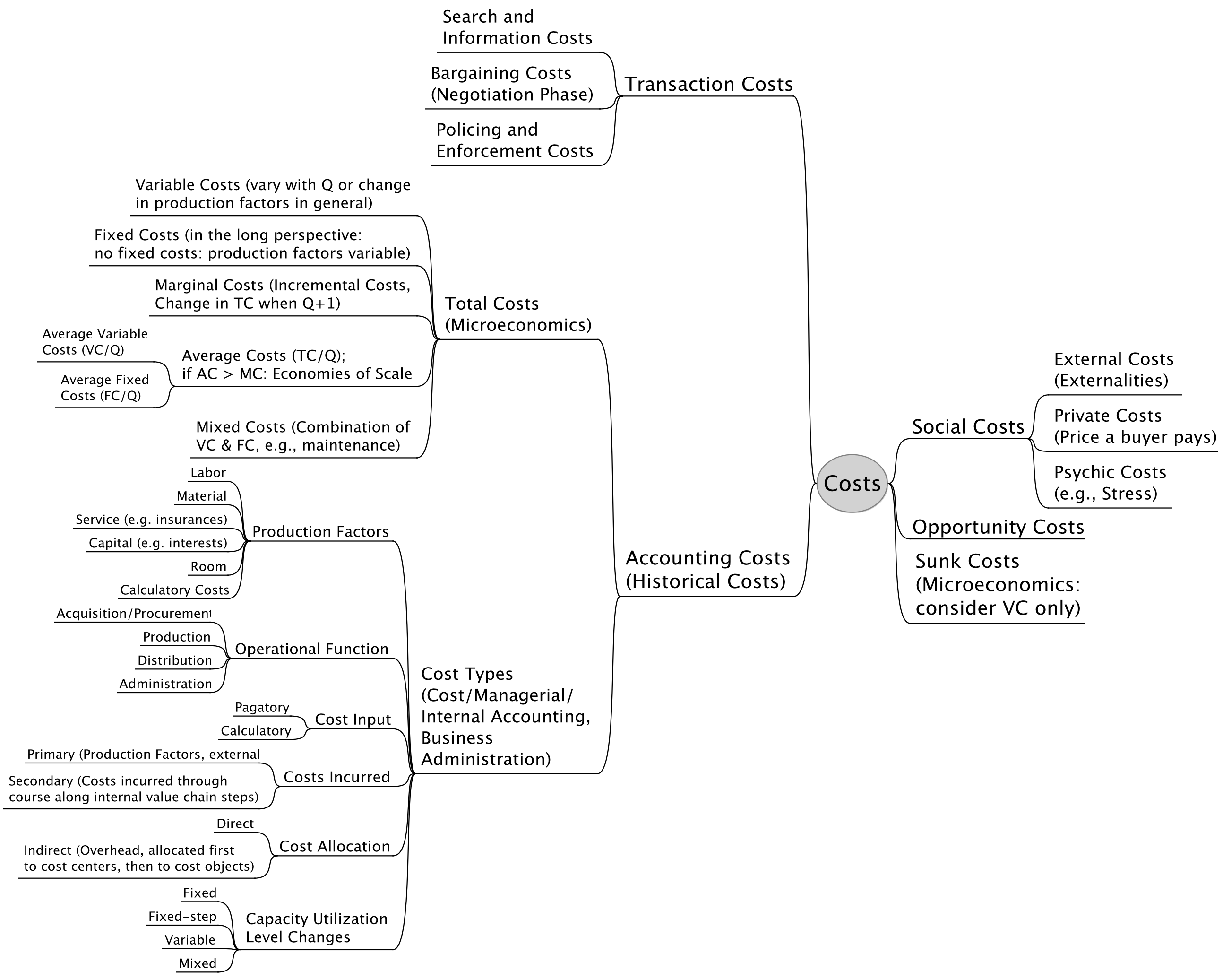
Costs
Transaction Costs
Search and Information Costs
Bargaining Costs (Negotiation Phase)
Policing and Enforcement Costs
Total Costs (Microeconomics)
Variable Costs (vary with Q or change in production factors in general)
Fixed Costs (in the long perspective: no fixed costs: production factors variable)
Marginal Costs (Incremental Costs, Change in TC when Q+1)
Average Costs (TC/Q); if AC > MC: Economies of Scale
Average Variable Costs (VC/Q)
Average Fixed Costs (FC/Q)
Mixed Costs (Combination of VC & FC, e.g., maintenance)
Accounting Costs (Historical Costs)
Cost Types (Cost/Managerial/ Internal Accounting, Business Administration)
Production Factors
Labor
Material
Service (e.g. insurances)
Capital (e.g. interests)
Room
Calculatory Costs
Operational Function
Acquisition/Procurement
Production
Distribution
Administration
Cost Input
Pagatory
Calculatory
Costs Incurred
Primary (Production Factors, external
Secondary (Costs incurred through course along internal value chain steps)
Cost Allocation
Direct
Indirect (Overhead, allocated first to cost centers, then to cost objects)
Capacity Utilization Level Changes
Fixed
Fixed-step
Variable
Mixed
Social Costs
External Costs (Externalities)
Private Costs (Price a buyer pays)
Psychic Costs (e.g., Stress)
Opportunity Costs
Sunk Costs (Microeconomics: consider VC only)

Figure 8: Overview of Cost Parameters[1]

1.AC denotes Average Costs, FC denotes Fixed Costs, MC denotes Marginal Costs, TC denotes Total Costs, VC denotes Variable Costs, Q denotes in microeconomics the output achieved.

In a wider view these 5 categories can be refined in the context of economic management tasks with the two following cost parameters:

6 **Audit Management Costs:**
These costs are required for a compliance check of an SLA or the verification of a contractual fulfillment of a pre-negotiated contract, typically an SLA in the networking domain. These costs could be considered, if required, as special fault management costs for external errors.

7 **Contract Management Costs:**
These costs cover contract negotiations and interactions with customers, providers, and peers, which may also include call centers as external interfaces to a network management system. They could be considered, if required, as special configuration management costs for setting up a bilateral or multilateral customer/user to provider relation.

Finally, in a much longer term time-scale the following cost parameter may become relevant:

8 **Research Costs:**
These costs exist for the design and development of network services in a determined networking technology environment, for a given market segment, and a class of customers.

Categories 1 to 5 show key characteristics of overhead costs. Network management cost elements usually are not directly allocatable to a specific cost object. In this context, cost objects consist in electronic services that are delivered to service consumers. Thus, in other words, network management costs cannot be attributed directly to a given service delivery, since this specific service delivery cannot be taken as the single or at least the main reason that network management tasks occur in the first place. However, a certain share in overall network management costs might consist of direct costs. This is the case if a network management activity is triggered by a given service delivery. The costs related to that specific network management activity are considered as direct costs. From the range presented in Figure 8, other cost type classifications might apply to network management costs in parallel, such as fixed or secondary costs. Category 8, research costs, is characterized in the same way as network management costs mainly by overhead/indirect costs.

Categories 6 and 7 are included in the transaction cost theory (cf. Figure 8). Contract management (category 7), however, goes beyond pure negotiations. It involves also costs for investigation, searching and — after a contract was concluded — costs incurred when contract determinations need to be ensured/enforced. Thus, audit-related and negotiation-related costs can be seen as a sub category of transaction costs. Again, this cost type classification need not to be seen as the only possible classification.

In order to provide an initial sample list of economic measures and cost parameters Figure 9 addresses for two examples a non-closing subset of parameters, measures, and charges relevant for the pure IP access service for private or home use — the most basic service to be thought of — and a VoIP service for corporate customers — a special VAS of large interest.

Finally, the set of open problems to be tackled in future research needs to address this type of mapping function between SLA and cost parameters — under certain assumptions — and needs to show in which cases a most generic approach of a sensitivity analysis can be developed in terms of a toolbox or a guideline in support of a provider and, at least

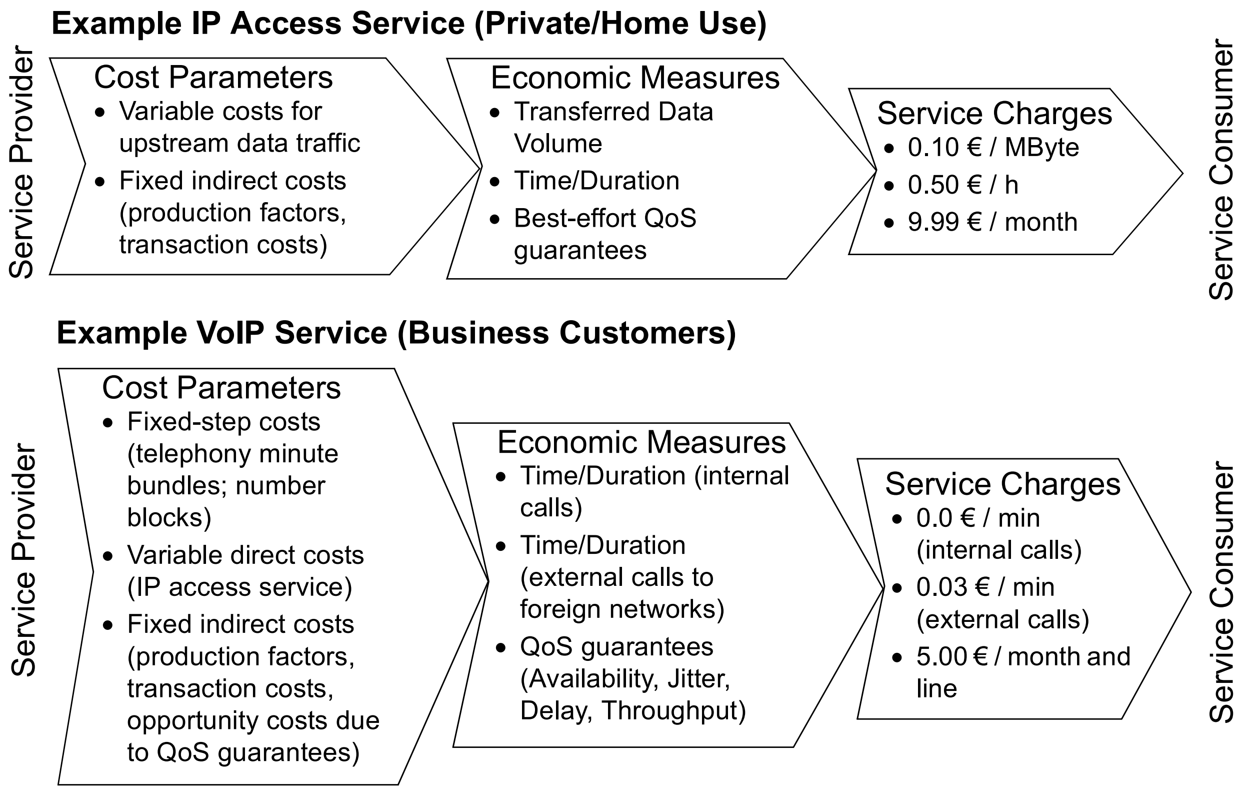
Example IP Access Service (Private/Home Use)
Service Provider
Cost Parameters
• Variable costs for upstream data traffic
• Fixed indirect costs (production factors, transaction costs)
Economic Measures
• Transferred Data Volume
• Time/Duration
• Best-effort QoS guarantees
Service Charges
• 0.10 € / MByte
• 0.50 € / h
• 9.99 € / month
Service Consumer
Example VoIP Service (Business Customers)
Service Provider
Cost Parameters
• Fixed-step costs (telephony minute bundles; number blocks)
• Variable direct costs (IP access service)
• Fixed indirect costs (production factors, transaction costs, opportunity costs due to QoS guarantees)
Economic Measures
• Time/Duration (internal calls)
• Time/Duration (external calls to foreign networks)
• QoS guarantees (Availability, Jitter, Delay, Throughput)
Service Charges
• 0.0 € / min (internal calls)
• 0.03 € / min (external calls)
• 5.00 € / month and line
Service Consumer

Figure 9: Cost Parameters, Economic Measures, and Service Charges for Two Sample Services

hypothetically, for business customers, who would like to evaluate a number of counter offers for their dedicated demands. Furthermore, a potential list of mandatory parameters of interest for a service or service class as well as optional ones may be determined based on the investigations of parameter dependencies and correlations.

4.2.2 Mobile Micro-payment and Accounting Systems

Micro-payments define the set of technology, which combines the action of payment — the transfer of value or a monetary claim from a payer to a payee — with the amount of such values, which are very small. Typically, micro-payment systems belong to the class of electronic payment systems, in which a type of e-money will be transferred, initiated, processed, and acknowledged fully electronically. Examples of micro-payment systems include Wallie, Minitix, and Way2Pay (cf. [40] for further details and systems). This type of e-money [4] — or electronic money — is defined by a complete electronic medium, in which this medium is issued by banks and exchanged in a fully electronic manner. Finally, the mobility aspect of those payment systems is added in case of

- the mobile device determining the payment instrument,
- the mobile device determining the payment terminal, or
- the mobile network determining the direct payment channel.

Thus, currently the following three systems "Paiement CB sur mobile", Paybox [42], and Moxmo show examples of mobile payment systems, not necessarily micro-payment systems, though. A nice overview on all aspects of payment systems, micro aspects, and technologies can be obtained from [40].

The mobility aspect of accounting systems (for the set of existing accounting technologies refer to Section 4.1.4), however, remains at this stage a preliminary one. This is based on the fact that a number of decisions have to be taken in advance. Initially, the location of the accounting data base is essential for either the reliable access to it as well as for an performance-sensitive access. Furthermore, the size of such an accounting data base is highly relevant for the type of device and its storage capabilities. Finally, the security concern of the operator utilizing a mobile accounting system has to be met, of course, meeting customers' demands at the same time.

Note that there is a fundamental difference between the security needed for "traditional" accounting systems and mobile micro-payment systems. Within traditional accounting systems, using for example protocols like RADIUS [45], Diameter [3], or NetFlow [7], information is exchanged between trusted parties. However, within micro-payment systems information is exchanged between parties that have no direct trust relationship. As a consequence, security aspects like data-integrity and non-repudiation are essential for such systems. Trust not only requires technical measures, but need to be enforced also by legislation.

For the design of an efficient and technically scalable mobile micro-payment system a set of system and security requirements, application specifics, and performance considerations have to be taken into consideration. In addition, such systems should support anonymity, be easy to use, and allow payments to be made across country and payment operator borders. Standardization and interoperability are, therefore, key and important requirements [41].

4.2.3 Service Provisioning Templates for IP QoS Delivery

This section describes the business relationships between principal actors for an IP QoS delivery. Two different possible business models are distinguished. The first one is largely in-line with the business model specified in the context of the EU MESCAL project [34], [25]. The second business model is based on observations of pure service providers recently emerged in the Internet, such as Skype.

Each of those two examples are described in terms of service provisioning templates, expressing the fact that the business model defines the product to be offered — here the service to be offered — and the concrete interdependencies of roles distinguished — here service users and customers as well as the three different provider roles — explain the interaction template those roles will typically show for the type of service under consideration.

4.2.3.1 Service Provisioning Template 1 — MESCAL-compliant

Most today's telecommunication companies that provide Internet access/services have the functionality of both ICPs and ASPs, meaning that they not only own the IP network infrastructure but also provide various services in a higher level. Figure 10 shows the template that assumes the integration of ICPs and ASPs. This means that individual end-customers not only are connected to the ICP but also subscribe to the services they are providing. The main focus is on the relationships between end-customers and ICP/ASPs and between ICP/ASPs themselves.

It is assumed that the interactions between ICPs and Access Providers for connecting end-customers to the IP network infrastructure owned by the ICPs either do not exist (end-customers are connected through means/facilities provided by the ICPs themselves), or if they do, they are completely orthogonal to (they do not affect) the provisioning of services; as such they are outside the scope of investigation.

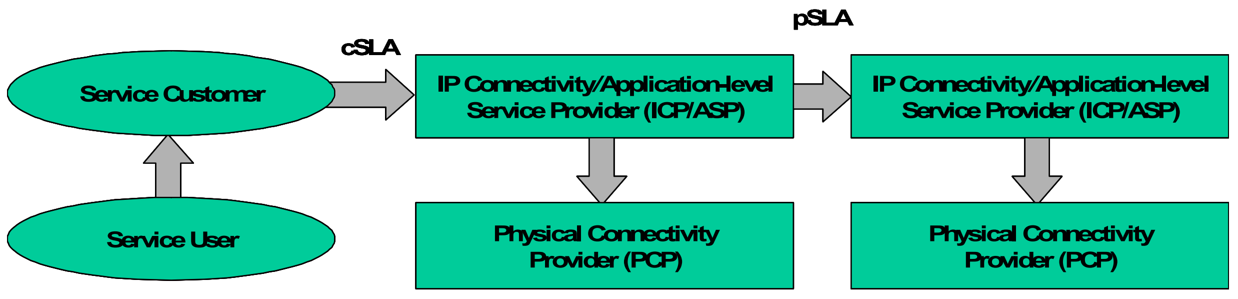
cSLA
pSLA
Service Customer
Service User
IP Connectivity/Application-level Service Provider (ICP/ASP)
IP Connectivity/Application-level Service Provider (ICP/ASP)
Physical Connectivity Provider (PCP)
Physical Connectivity Provider (PCP)

Figure 10: Service Provisioning Template 1 — MESCAL-compliant

In the scenario shown above, the primary interest is in the investigation of Service Level Agreements (SLA) underlying the interactions between the stakeholder domains of concern from technical perspectives (*e.g.*, specifications, protocols) and the required functionality in the ICP/ASP domains for supporting these SLAs. Regarding the latter, the emphasis is on SLA fulfilment aspects through traffic engineering and service admission control functions. More specifically, we are interested in the technical aspects of SLAs, referred to as SLSs (Service Level Specifications). Service level specification can be described as a documented result of negotiation that specifies availability, performance, pricing and other aspects of a transport service a provider offers to a customer. The terms customer SLS (cSLS) and provider SLS (pSLS) are used to denote respectively the SLS' between end-customers and ICP/ASPs, and among peering ICP/ASPs themselves.

Customer SLS (cSLS)

A cSLS is the technical specification of the SLA between the end-customer and the ICP/ ASP. It basically specifies how the end-customer traffic should be treated within the ICP/ ASP's network. The intra-domain cSLS can be taken as an example, where traffic source and destination are can be covered by one single provider. A typical cSLS includes the following elements:

- SLS Id:
  SLS ID identifies uniquely the SLS specific to the end-customer;
- Scope:
  As far as a single ICP/ASP network is concerned, there exist three types of SLS scope for end-customers who would inject traffic:
  - The pipe model (1, 1) has one ingress and one egress
  - The hose model (1, N) has one ingress and multiple egresses
  - The one-to-any model (1, *) has one ingress and a wildcard with egresses
- Service Schedule:
  Service schedule specifies when the cSLS is active, i.e. operating times such as per-day or per-week
- Flow descriptor:
  Flow descriptor is essentially the packet stream identifier that includes source address, port number and DSCP (Differentiated Services Code Point).
- QoS parameter:
  Include, *e.g.*, bandwidth, delay, jitter, packet loss ratio.

- Traffic Descriptor:
  A typical example is a token bucket algorithm for identifying in- and out-of-profile packets.
- Excess Treatment:
  Excess treatment indicates how to treat out-of-profile packets, *e.g.*, dropping, shaping, or remarking.
- Service availability:
  Service availability means the percentage of time the service is available, *e.g.*, 99.9%.

Provider SLS (pSLS)

As mentioned earlier, individual ICP/ASPs need to establish provider level SLS' with each other if they would like to expand IP reachability or higher level services to other providers. With the presence of pSLS' between multiple ICP/ASPs, Customers are able to subscribe and use services that are provided by remote providers in the Internet.

In essence, pSLSs extend, typically to the end of QoS traffic exchange, the respective agreements that exist today between the providers in the best-effort Internet -for transiting or exchanging traffic. As such, they should be inline with the specific context of traffic exchange, which is implied by the particular business relationships holding between providers. Compared to cSLS', a significant difference is that the technical service agreement (pSLS) is based on aggregated traffic other than individual flows. Put in other words, the task of negotiating pSLS' between ICP/ASPs is to consider how to fulfil individual cSLS' from customers in an aggregate fashion.

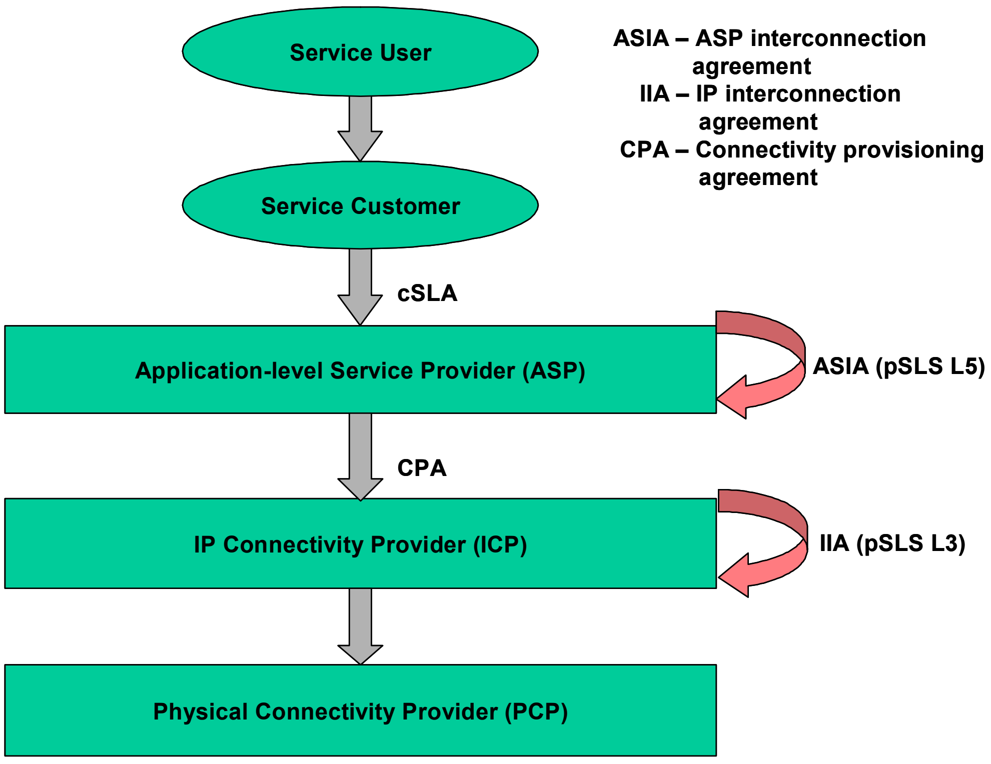
Service User
ASIA – ASP interconnection agreement
IIA – IP interconnection agreement
CPA – Connectivity provisioning agreement
Service Customer
cSLA
Application-level Service Provider (ASP)
ASIA (pSLS L5)
CPA
IP Connectivity Provider (ICP)
IIA (pSLS L3)
Physical Connectivity Provider (PCP)

Figure 11: Service Provisioning Template 2 — Pure Service Provider-compliant

4.2.3.2 Service Provisioning Template 2 — Pure Service Provider-compliant

In this template, the functionality of ASPs and ICPs are distinct. Specifically, ASPs provide higher level services to customers without owning any IP layer infrastructure. On the other hand, ICPs only offer IP connectivity services to ASPs for supporting higher level applications and they do not interact directly to customers. Take Skype as an example: It is able to provide IP telephony with only VoIP gateways at the edge of INP's networks, but does not necessarily have a physical IP network.

Today, services provided by Skype are based on the best effort Internet without guaranteeing any QoS. In order to offer IP telephony services with QoS guarantees, ASPs need IP level QoS support (*e.g.*, dedicated bandwidth resources) from ICPs. ASPs interact vertically with ICPs on the basis of Connectivity Provisioning Agreements (CPA), as shown in Figure 11. Beyond the forwarding and the QoS treatment of the service control and data traffic entering the ICP's network from the ASP's sites, the ICP offers to the ASP means to control the connectivity provisioning, such as receiving reports and alarms, invoking admission control at the IP network resources level, or enforcing the required configuration on the INP's network on service activation.

Similar to the first business model, for expanding the scope and augmenting the portfolio of the offered services, ASPs may interact horizontally with each other through setting up business agreements called ASP Interconnection Agreements (ASIA), which are also called pSLS layer 5 (Session Layer). On the other hand, the underlying ICP also need layer 3 pSLS between each other in order to expand their IP reachability services. This type of pSLSs is called IP Interconnection Agreements (IIA). The elements to be included in ASIAs and IIAs are still under study at this moment.

4.3 Integration Steps Foreseen

Those two sets of mechanisms indicated in Section 4.1 and Section 4.2 are required to be combined to achieve an economic management approach for network services. Due to those detailed lists of parameters — originating from the QoS and RADIUS, Diameter, and general accounting world as well as from the cost and economics measures domain — the key interfaces between them have been initially identified. Of course, this step determines the first one required, however, this basis clarified views and abstracted away from low-level details, which will not be required. Thus, the integration module — as indicated in Figure 6 — will become a key functional aspect of future economic management solutions.

The second step envisioned is addressed already in the continuation work of Task 8.1 and Task 8.2 of WP8: The refinement of SLA management tasks, the monitoring of SLAs, the investigation of promise theory for network contracts, the consolidation of accounting work for various advanced — grid and broadband — services, investigations on cost parameters and tariffs, the analysis of models for multi-domain scenarios, and the flexible design of bandwidth trading schemes.

5 Preliminary Conclusions

The paradigm shift to Information Technology (IT) service management currently seen is a consequence of various trends such as the liberalization of the telecommunication market, the convergence between computer networks, and telecommunication systems and the need for an end-to-end management. New technical, organizational, and economic challenges become apparent and new technologies appear to be full of promise. While the domains of traditional network management and IT management have been considered separately so far, the interrelation and inter-operation of services from both domains does require a joint approach.

Managing new services on many different types of networks with many different players are among those challenges. Along with them is the need for the provisioning of a seamless interoperability and interdependence between services and networks. In this area, service providers compete for customers with high quality services, with the ability to deploy services rapidly and efficiently, to make the ordering of services extremely simple, to keep inconvenience for the customer at a minimum, and to provide for the ability to rapidly introduce and roll out new service offerings in various locations and systems. Instead of resources, such as network devices, end systems, and applications, it is necessary to think in terms of services and service quality.

Therefore, the concept of such a service is a recent advance in the understanding of networking technology as well as information technology. But what is such a service in detail, what is a Service Level Agreement (SLA), and what are respective management mechanisms for those? In order to analyze these questions in a much greater level of detail several key models and terms have been discussed first to gain a common basis of understanding. Besides the definition of many terms, the specification of the key characteristics and features of the

- Deployment Model,
- Business Model,
- Role Model,
- Service Model, and
- Charging Model

have been discussed and agreed upon. By doing this, a common basis for further investigations, technical and economic discussions, and developments has been defined.

Based on this common ground, the next research and collaboration issue was dedicated to the investigation of key mechanisms, including technical as well as economic management mechanisms. Those technical mechanisms were devoted to the key research issues such as formalizing SLAs, SLA negotiation mechanisms, policy-based management, and accounting. Discussing about technical aspects of SLAs and accounting has lead to the research issue of respective economic mechanisms. Of special importance was the definition of economic measures and cost parameters, mobile micro-payment schemes, accounting, and the analysis of service provisioning templates in order to move from “bits to business values”.

Due to the efficient research collaboration between the partners involved done so far, a number of research results have been achieved already, as demonstrated in this deliverable as well as the named selected cooperation papers found below. Those results verify the teamwork started between EMANICS WP8 partners and determine a commitment for further joint research achievements.

6 Glossary

This section outlines again the major terms which form the basis of the key models of an economic management approach and which refer to mechanisms of an economic service management infrastructure described in this document.

Accounting

Accounting defines the process of keeping and aggregating information about the amount of service units used by a service consumer, *e.g.*, the amount of Mega Byte (MB) downloaded or the number of Central Processing Unit (CPU) cycles used.

Business Model

A Business Model is the formal definition of a business concept. It defines offered services, roles of participants, and their interactions.

Charging

Charging is the process of applying a specific tariff specified for a particular session to the accounting information collected during the use of the service. The result of a charging process is a calculated charge that the service consumer has to pay to the service provider in return for providing the service.

Cost Parameter

Cost parameters define different types of costs involved in a service offered, such as transaction costs and accounting costs, which are complemented by social costs, opportunity costs and sunk costs.

Deployment Model

A deployment model defines the instantiation of a business model into operation. Thus, different application domains or application areas will show different deployment models.

Economic Measure

Economic measures define the range of potential quantified metrics applicable to a service, typically technical values derived from, *e.g.*, QoS parameters or metering parameters.

End-to-end

The attribute end-to-end defines the scope of a connection's property to be the entire connection from its starting point (typically an Service Access Point (SAP) to a layer 7 service defined by the Connection End Point (CEP) of the corresponding application process) to its destination (defined by an SAP and its respective CEP).

Policy

A policy defines a course or method of action selected among alternatives and in light of given conditions to guide and determine present and future decisions.

Pricing

Pricing is the process of determining the price or tariff for the use of a service typically performed by a service provider. This process includes the collection of information from different sources which are needed to determine the price, depending on the pricing strategy that is applied, *e.g.*, cost-based pricing or market-based pricing.

Quality-of-Service

Quality-of-Service (QoS) specifies the performance at which a service is provided to a service consumer, based on specific quantitative performance parameters, *e.g.*, throughput, delay, or jitter, which have an impact on the quality perceived by the service consumer, such as good, fair, or poor.

Role

According to promise theory, a role is associated with types of promises, i.e. categories of promise bodies. In this respect, a collection of agents can be associated with a role, given that the members of the collection make (or receive) the same type of promise to (from) some third party.

Service

A service is the entity or unit of work offered by a service provider on behalf of a service consumer who can use the service. In general, a service includes several types of resources, including hardware- and software resources such as computing power, network links, storage capacity, and content, and it may even be composed of several sub-services [20], [23].

Service Charge

A service charge is defined in terms of monetary units per service unit utilized, which is typically mapped from QoS parameters or metering parameters.

Service Level Agreement

A Service Level Agreement (SLA) defines the terms under which a service is offered to a service customer at a specific Service Access Point (SAP). The SLA includes a set of parameters which specify the service and the QoS under which it is provided (*e.g.*, the amount of bandwidth allocated, the involved session partners, metrics and algorithms that are used to compute SLA parameters), accountable units and the tariff which is used to charge for the service usage. Besides, several other aspects such as penalties or actions, respectively, to be taken if SLA objectives (*i.e.,* guarantees) are violated, trust relationships are part of a SLA.

Service Level Management

Service Level Management (SLM) is the process of specifying and monitoring services in a standardized fashion according to an SLA. This includes the collection of service user requirements, the definition and planning of services and SLAs, as well as the monitoring of service parameters specified in the SLA.

Service Provisioning Template

A service provisioning template expresses the fact that a business model defines the product or service to be offered and the concrete interdependencies of roles distinguished, *e.g.*, service users and customers as well as different provider roles, explain the interaction template those roles will typically show for the type of service under consideration.

Service Reliability

Service Reliability is the inverse of the probability for service failure. Failure may be due to a service execution error or complete service unavailability.

Session

A session defines the use of a particular service, *e.g.*, the download of a file or the use of some amount of computing power. A session has always at least two session partners, a service provider and a service consumer. In specific scenarios, the number of service providers and service consumers may also be higher [23].

Tariff

A tariff defines how service usage needs to be accounted and charged for. It is represented by a specific tariff formula and a set of tariff parameters which need to be agreed upon by both session partners. For example, a volume-based tariff could be used to account and charge for a file download. A tariff may also contain further parameters that have to be computed dynamically during a session, *e.g.*, depending on the time of day [23].

7 References

[1] M. Burgess: *An Approach to Understanding Policy Based on Autonomy and Voluntary Cooperation;* IFIP/IEEE 16th International Workshop on Distributed Systems Operations and Management (DSOM'06), Barcelona, Spain, October 2006, Lecture Notes in Computer Science LNCS, Springer, Heidelberg, Vol. 3775, pp 97-108.

[2] M. Burgess, S. Fagernes: *Pervasive Computing Management: A Model of Network Policy with Local Autonomy;* Submitted to IEEE Transactions on Software Engineering, 2006.

[3] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, *Diameter Base Protocol;* Internet Engineering Task Force (IETF) RFC 3588, September 2003.

[4] L.J. Camp, M. Sirbu, J.D. Tygar: *Token and Notational Money in Electronic Commerce*; 1st USENIX Workshop on Electronic Commerce, New York, U.S.A., July 1995.

[5] A. Chavez and P. Maes, Kasbah: *An Agent Marketplace for Buying and Selling Goods;* 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, U.K., April 1996, pp 75-90.

[6] H. Chesbrough, R. S. Rosenbloom: *The Role of the Business Model in Capturing Value from Information: Evidence from Xerox Corporation's Technology Spin-off companies*; Industrial and Corporate Change, Vol. 11, No. 3, 2002, pp. 529-555.

[7] Cisco Systems: *Cisco IOS Netflow: Introduction;* http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html, May 2006.

[8] B. Claise (Edt.), *IPFIX Protocol Specification;* IPFIX Working Group, Internet Engineering Task Force (IETF), draft-ietf-ipfix-protocol-21.txt, April 2006.

[9] CORBA Services Complete Book: *OMG Specification formal/98-12-09;* Object Management Group, December 1998.

[10] N. Damianou, N. Dulay, E. Lupu, M. Sloman: *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, The Language Specification Version 2.3;* IC Research Report DoC 2000/1, Imperial College of Science, Technology and Medicine, University of London, Department of Computing, October 2000.

[11] V. Danciu, B. Kempter: *From Processes to Policies — Concepts for Large Scale Policy Generation;* Managing Next Generation Convergence Networks and Services: Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS’04), Seoul, Korea, April 2004.

[12] G. Dreo Rodosek: *A Framework for IT Service Management;* Habilitation Thesis, Ludwig Maximilian Universität, Munich, Germany, 2002.

[13] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry: *The Common Open Policy Service (COPS) Protocol;* Internet Engineering Task Force (IETF) RFC 2748, January 2000.

[14] R. Edell, P. P. Varaiya: *Providing Internet Access: What we learn from the INDEX trial;* Keynote Talk at Infocom’99 New York, INDEX Project Report #99-010W. URL: http://www.index.berkeley.edu/99-010W, March 1999.

[15] ETSI TR 121 905 V6.10.0 (2005-09), Digital cellular telecommunications system (Phase 2+): *Vocabulary for 3GPP Specifications;* 3GPP TR 21.905 Ver. 6.10.0 Release 6, 2005.

[16] FIPA: *The Foundation of Intelligent Physical Agents;* IEEE FIPA Standards Committee, http://www.fipa.org/, June 2006.

[17] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, M. Langer, M. Nerb, I. Radisic, H. Roelle, H. Schmidt: *Towards Generic Service Management Concepts — A Service Model Based Approach;* 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001), Seattle, Washington, USA, May 2001.

[18] M. Garschhammer, R. Hauck, B. Kempter, I. Radisic, H. Roelle, H. Schmidt: *The MNM Service Model — Refined Views on Generic Service Management;* Journal of Communications and Networks, December 2001.

[19] GEMSS Project: *Grid-Enabled Medical Simulation Services;* 5th Framework EU Project, IST Program, No. IST-2001-37153, http://www.gemss.de, February 2005.

[20] J. Gerke (Editor): *Peer-to-peer Services Architecture;* D5 MMAPPS Project Deliverable, September 2004.

[21] J. Gerlach, B. Neumann, E. Moldauer, M. Argo, D. Frisby: *Determining the Cost of IT Services*; Communications of the ACM, Vol. 45, No. 9, September 2002, pp. 61-67.

[22] GSM World: *Tapping the Potential of Roaming;* http://www.gsmworld.com/using/billing/potential.shtml, June 2006.

[23] D. Hausheer: PeerMart: *Secure Decentralized Pricing and Accounting for Peer-to-Peer Systems;* Dissertation ETH Zurich No. 16200, Shaker Verlag, Aachen, Germany, March 2006.

[24] H.-G. Hegering, S. Abeck, B. Neumair: *Integriertes Management vernetzter Systeme - Konzepte, Architekturen und deren betrieblicher Einsatz;* dpunkt-Verlag, ISBN 3-932588-16-9, 1999.

[25] M. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, H. Asgari and P. Georgatsos: *Provisioning for Inter-domain Quality of Service: The MESCAL Approach*; IEEE Communications Magazine, Vol. 43, No. 6, June 2005, pp 129-137.

[26] P. C. K. Hung, H. Li, J.-J. Jeng: *WS-Negotiation: An Overview of Research Issues;* 37th Annual Hawaii International Conference on System Sciences (HICSS'04), Hawaii, U.S.A., January 2004.

[27] Information Technology - Open Systems Interconnection - *Systems Management - Management Functions,* ISO 10164-x, International Organization for Standardization and International Electrotechnical Committee, 1991-97.

[28] IPDR.org, *IPDR/XML File Encoding Format;* Version 3.5.0.1, November 2004.

[29] R. S. Kaplan, A. A. Atkionson: *Advanced Management Accounting;* 3rd Edition, Prentice Hall, Upper Saddle River, U.S.A., January 1998.

[30] P. Kurtansky, B. Stiller: *State of the Art Prepaid Charging for IP Services;* 4th International Conference on Wired/Wireless Internet Communications (WWIC 2006), May 10-12, 2006, Bern, Switzerland, Lecture Notes in Computer Science LNCS, Springer, Heidelberg, Vol. 3970.

[31] C. de Laat, G. Gross, L. Gommans, *Generic AAA Architecture;* Internet Engineering Task Force (IETF) RFC 2903, August 2000.

[32] H. Ludwig, A. Keller, A. Dan, R. P. King, R. Franck: *Web Service Level Agreement (WSLA) Language Specification;* IBM T.J. Watson Research Center, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, 2003.

[33] B. Moore, E. Ellesson, J. Strassner, A. Westerinen: *Policy Core Information Model - Version 1 Specification;* Internet Engineering Task Force (IETF) RFC 3060, February 2001.

[34] P. Morand, M. Boucadair, T. Coadic, P. Levis, R. Egan, H. Asgari, D. Griffin, J. Griem, J. Spencer, M. Howarth, N. Wang, S. Georgoulas, K.H. Ho, P. Flegkas, P. Trimintzios, G. Pavlou, P. Georgatsos, E. Mykoniati, I. Liabotis, T. Damilatis: *Final specification of protocols and algorithms for inter-domain SLS management and traffic engineering for QoS-based IP service delivery;* D1.3 MESCAL Deliverable, http://www.mescal.org/, June 2005.

[35] M3I: *Market Managed Multi-service Internet;* 5th Framework EU Project, IST Program, No. 11429, URL: http://www.m3i.org, January 2000.

[36] T. Nolle: *A New Business Layer for IP Networks;* Business Communications Review, http://www.ipsphereforum.org/newsevents/07nollereprint.pdf, July 2005.

[37] OASIS: *Automated Negotiation of Collaboration-Protocol Agreements Specification;* ebXML Collaboration Protocol Profile and Agreement Technical Committee, http://www.oasis-open.org/committees/ebxml-cppa/negotiation, September 2003.

[38] A. Odlyzko: *Internet pricing and the history of communications;* Computer Networks, Vol. 36, 2001, pp 493-517.

[39] A. Osterwalder: *The Business Model Ontology: A Proposition in a Design Science Approach*; Ph.D. Thesis, University of Lausanne HEC, 2004.

[40] R. Párhonyi: *Micro-payment Gateways;* University of Twente, CTIT Ph.D. Thesis Series, No. 05-72, Enschede, The Netherlands, October 2005.

[41] R. Párhonyi, L.J.M. Nieuwenhuis, A. Pras: *Second Generation Micro-payment Systems: Lessons Learned,* 5th IFIP conference on e-Commerce, e-Business, and e-Government (I3E 2005), Poznan, Poland, October 2005 (also published in the Payments and Settlements News No.24 prepared by the ePSO team at the European Central Bank).

[42] Paybox Solutions AG: *Mobilizing People — Paybox;* http://www.paybox.de/, May 2006.

[43] M. E. Porter: *Competitive Strategy: Techniques for Analyzing Industries and Competitors*; 1st Edition, Free Press, New York, U.S.A., June 1998.

[44] P. Reichl, P. Flury, J. Gerke, B. Stiller: *How to Overcome the Feasibility Problem for Tariffing Internet Services: The Cumulus Pricing Scheme;* IEEE International Conference on Communications (ICC 2001), Helsinki, Finland, June 2001, pp 2079–2083.

[45] C. Rigney, *RADIUS Accounting;* Internet Engineering Task Force (IETF) RFC 2866, June 2000.

[46] C. Rigney, S. Willens, A. Rubens, W. Simpson, *Remote Authentication Dial In User Service (RADIUS);* Internet Engineering Task Force (IETF) RFC 2865, June 2000.

[47] Service Level Agreement and SLA Guide — *The SLA Toolkit:* http://www.service-level-agreement.net/, May 2006.

[48] R. Smith, R. Davis: *The contract Net protocol: High level communication and control in a distributed problem solver.* IEEE Transactions on Computers 1980; Vol. 29, No. 12, pp 1104-1113.

[49] B. Stiller: *A Survey of Charging Internet Services;* in S. Aidarous, T. Plevyak (edts.): Managing IP Networks, Challenges and Opportunities, Wiley Interscience, 2003

[50] B. Stiller, T. Braun, M. Günter, B. Plattner: *The CATI Project: Charging and Accounting Technology for the Internet;* 5th European Conference on Multimedia Applications, Services, and Techniques (ECMAST'99), Madrid, Spain, May 26-28, 1999, Lecture Notes in Computer Science LNCS, Springer Verlag, Heidelberg, Vol. 1629, pp 281-296.

[51] B. Stiller, P. Reichl, S. Leinen: *Pricing and Cost Recovery for Internet Services: Practical Review, Classification, and Application of Relevant Models;* Netnomics – Economic Research and Electronic Networking, Vol. 3, No. 2, September 2001, pp 149–171.

[52] T. M. T. Nguyen, N. Boukhatem: *Policy-based Service Level Negotiation with COPS-SLS;* Conference on Network Control and Engineering, (NetCon'03), Muscat, Oman, October 14-15, 2003.

[53] D. C. Verma: *Policy-based Networking;* Technology Series, New Riders Publishing, Indianapolis, Indiana, 2000.

[54] J. Walker, A. Kulkarni (Edts.): *Common Open Policy Service (COPS) Over Transport Layer Security (TLS);* Internet Engineering Task Force (IETF) RFC 4261, December 2005.

[55] E. Wustenhoff: *Service Level Agreement in the Data Center;* Sun, http://www.sun.com/blueprints/0402/sla.pdf, April 2002.

[56] Y. Yemini: *The OSI Network Management Model;* IEEE Communications Magazine, Vol. 31, No. 5, pp 20-29, May 1993.

(This page was left blank intentionally.)

8 Abbreviations

3G	3rd Generation
3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization, and Accounting
ABC	Activity-based Costing
AC	Average Cost
API	Application Programming Interface
AR	Accounting Record
ASIA	ASP Interconnection Agreements
ASP	Application-level Service Provider
AVP	Attribute Value Pairs
B2B	Business-to-Business
B2C	Business-to-Consumer
BR	Billing Record
CDC	Caisse des Dépôts et Consignations
CDMA	Code Division Multiple Access
CETIM	University of Federal Armed Forces Munich (Universität der Bundeswehr)
CEP	Connection End Point
COPS	Common Open Policy Service Protocol
CORBA	Common Object Request Broker Architecture
CPA	Connectivity Provisioning Agreements
CPA	Collaboration Protocol Agreement
CPP	Collaboration Protocol Profile
CPS	Cumulus Pricing Scheme
CPU	Central Processing Unit
cSLA	Customer Service Level Agreement
cSLS	Customer Service Level Specification
CSM	Customer Service Management
CR	Charging Record
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DVO	Dynamic Virtual Organization
EMANICS	Management of the Internet and Complex Services
EU	European Union
EV-DO	Evolution-Data Optimized
FC	Fixed Cost
FCAPS	Fault, Configuration, Accounting, Performance, Security
FIPA	Foundation for Intelligent Physical Agents
FR	Frame Relay
GEMSS	Grid-enabled Medical Simulation Services

h	Hour
HIO	Oslo University College
IC	Imperial College
ICMP	Internet Control Message Protocol
ICP	Internet Connectivity Provider
ICT	Information and Communication Technology
IDC	Internet Data Center
IETF	Internet Engineering Task Force
IIA	IP Interconnection Agreements
INRIA	Institut National de Recherche en Informatique et Automatique
IP	Internet Protocol
IPDR	Internet Protocol Detail Record
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
IT	Information Technology
ITIL	IT Infrastructure Library
IUB	International University Bremen
KTH	Royal Institute of Technology Stockholm
LMU	Ludigws-Maximilian-Universität München
M	Mega
MB	Mega Byte
MC	Marginal Cost
MESCAL	Management of End-to-end Quality of Service Across the Internet at Large
min	Minute
NP	Network Provider
OASIS	Organization for the Advancement of Structured Information Standards
OLA	Operation Level Agreements
OSI	Open System for Interconnect
PCP	Physical Connectivity Provider
PDP	Policy Decision Point
PEP	Policy Enforcement Point
POTS	Plain Old Telephony System
P2P	Peer-to-peer
pSLA	Provider Service Level Agreement
pSLS	Provider Service Level Specification
PSNC	Poznan Supercomputing and Networking Center
PSTN	Public Switched Telephony Network
Q	Output (in microeconomics)
QoS	Quality-of-Service
RADIUS	Remote Authentication Dial-In User Service
RSVP	Resource Reservation Protocol

RT	Requirement Type
SAP	Service Access Point
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SLM	Service Level Management
SLS	Service Level Specification
SNMP	Simple Network Management Protocol
SRTCPU	Scarce Resource Trading for CPU
TAP	Transferred Account Procedure
TC	Total Cost
UMTS	Universal Mobile Telecommunication System
UN	Underpinning Contract
UniBwM	University of Federal Armed Forces Munich (Universität der Bundeswehr)
UniS	University of Surrey
UniZH	University of Zürich
UPC	Universidat Politecnica de Catalunya
UPI	University of Pitesti
UT	University of Twente
VAS	Value-added Service
VC	Variable Cost
VO	Virtual Organization
VoIP	Voice-over-IP
WLAN	Wireless Local Area Network
WP	Work Package
WSLA	Web Service Level Agreement
WS	Web Service
xDSL	Type x Digital Subscriber Line
XML	eXtended Mark-up Language

9 Acknowledgements

This deliverable was made possible due to the large and open help of the WP8 team of the EMANICS team within the NoE, which includes especially besides the deliverable authors as indicated in the document control, Thomas Bocek (UniZH) and Frank Eyermann (UniBwM) as well. Many thanks to all of them.

(This page was left blank intentionally.)

10 Selected Cooperation Work

This section of selected cooperation work covers a range of work started recently in the context of EMANICS WP8 as well as sometimes shortly before. Thus, the content in this section consists out of paper abstracts and summaries from single institutions (early starts) as well as joint work between EMANICS partners (recent starts).

To provide an overview of these areas of work, the following subsections list authors, abstracts, and the table of content, if they exist, or a respective sketch of the idea to be worked on in the next month of EMANICS. In case of a full paper being available, it is part of this deliverable at its final end, following the sequence as their summaries below.

Thus, D8.1 covers 2 single affiliation papers as well as 2 joint affiliation papers being ready and 3 joint affiliation papers being under preparation.

10.1 Uncertainty in Global Application Services with Load Sharing Policy

Authors: Mark Burgess and Sven Ingebrigt Ulland — HIO

Abstract:

With many organizations now employing multiple data centres around the world to share global traffic load, it is important to understand the effects of geographical distribution on service quality. The Domain Name Service (DNS) is an important component for global load balancing. Using controllable simulations, we show that wide area sharing can play an important role in optimization of response times when traffic levels exceed that which can be supplied by a local infrastructure. We compute the probability of being able to meet Service Level Objectives as a function of DNS caching policy (Time To Live), so that service providers can account for DNS error margins in Service Level Agreements.

Table of Contents:

- DNS performance and high level load balancing are defined
- Time to Live policy is discussed
- Experiments to verify the predictability of DNS based load balancing are performed
- Conclusions about the limitations of predictable service levels are drawn

10.2 Promise Theory on ITIL

Authors: HIO and UniBwM

A sketch of this paper exists and is planned as follows:

Table of Contents:

- Summary of Ideas from ITIL
- Background
- Basic Promise Theory
- Application of Promise Theory to ITIL
- Necessary and Sufficient Roles
- Implementing “Requirements”
- Data Support

- Continuity Management
- Coordination Management

10.3 Bandwidth Trading

Authors: David Hausheer[1], Burkhard Stiller[1], Aiko Pras[2] — [1] UniZH, [2] UT

Abstract:

The Internet is becoming increasingly popular as an electronic marketplace. As a result of the rapid technological progress, the near future will show a wide variety of goods such as computer resources and services being traded over the Internet by many buyers and sellers. This vast amount of goods and traders requires reliable trading mechanisms to be in place which are truly efficient and scalable.

At the same time, Peer-to-peer (P2P) networks are emerging as a new design approach for building scalable and fault-tolerant applications. Fully decentralized marketplaces based on P2P networks enable the trading of services over the Internet in a technically and economically feasible way. In this paper the existing technology for a decentralized auction-based marketplace supporting generic services is refined to meet the specific requirements of a market for trading bandwidth.

A specific bandwidth trading scenario is specified that shall be supported by the developed marketplace and from which detailed requirements are derived. The scenario is targeted at trading optical links over a fully virtualized networking infrastructure. The particular technical and economic aspects of such a bandwidth service are investigated and the specific service parameters defined. A fully decentralized market infrastructure is designed based on PeerMart's double auction mechanism allowing network operators to buy bandwidth services on demand, and to even sell them to other network operators if not used.

Table of Contents:

10.4 An Accounting Model for Dynamic Virtual Organizations

Authors: Matthias Göhner[1], Martin Waldburger[2], Fabian Gubler[2], Gabi Dreo Rodosek[1], Burkhard Stiller[2] — [1] UniBwM, [2] UniZH

Abstract:

The provisioning of remote and composed services in support of various application areas has emerged over recent times dramatically. Thus, the concept of Grids has evolved in the sense of a common platform for electronic service provisioning in multi-domain environments. While, traditionally, Grids have seen a quite static existence, many new

service compositions have to take place on-demand and for certain periods of time only. Thus, the concept of Virtual Organizations (VO) delivers a highly suitable representation of such dynamic Grids. However, an open problem at this stage can be seen in the lack of applicable, distributed, and efficient accounting schemes for commercial resource and service consumptions. Even for management purposes, *e.g.*, sampling or archiving, this functionality is essential.

Therefore, a comprehensive model for Grid accounting has been developed and suitable accountable units have been defined, in which an underlying activity- and resource-based accounting model covers economic cost theory. Furthermore, this work is based on a service model proposed for service provisioning in dynamic VOs.

Table of Contents:

10.5 System Administration and the Business Process

Authors: Mark Burgess — HIO, Comments from UniZH

Abstract:

The integration of computer technology into a business process is an obvious goal for an organization or company that relies on information services. In this work the following question is posed and replied to: Is it possible to understand business modelling and analytical system administration in the same light? In other words, how can someone bring a rational scientific method to bear on the problem of building a business around an Information Technology (IT) infrastructure, or building an IT infrastructure for an existing business? To do this, the process of business modelling has to be viewed in the same light as the process of modelling the rest of the information system, and to relate these ideas to the practical issues that businesses have to contend with.

Table of Contents:

10.6 Process-oriented Service Level Management

Authors: Thomas Schaaf[1], Stylianos Georgoulas[2] — [1] LMU, [2] UniS

Abstract:

Preliminary Abstract: Service Level Management (SLM) belongs to ITIL's (Information Technology Infrastructure Library) Service Delivery Processes and is thus of tactical and strategic importance for the IT organization as well as the core business of an enterprise. Since it is meant to represent the customer's demands towards the IT (Information Technology) organization and at the same time lead negotiations with the customer, the SLM process forms the interface between customer and IT organization on the one hand (role-related view) and between business processes and IT processes on the other hand (process-related view). In this work, we analyze the areas of conflict that derive from this situation and find vital operational requirements to implementing the SLM process, considering aspects of great significance for this process and its relation to other Service Delivery and Service Support processes.

10.7 Introducing CPU Time as a Scarce Resource in P2P Systems to Achieve Fair Use in a Distributed DNS

Authors: Thomas Bocek, David Hausheer, Reinhard Riedl, Burkhard Stiller — UniZH

Publication: 9th IEEE Global Internet Symposium 2006, Barcelona, Spain, April 2006.

Abstract:

Peer-to-peer (P2P) systems are flexible, robust, and self-organizing resource sharing infrastructures which are typically designed in a fully decentralized manner. However, a key problem of such systems are peers overusing a resource. This paper presents a fully decentralized scheme to achieve fair use in P2P systems, which does not require a priori information about a peer. The approach developed is based on a Scarce Resource Trading scheme (SRTCPU), which utilizes CPU (Central Processing Unit) time as a form of payment. SRTCPU provides an incentive to offer CPU time in return of consuming a scarce resource. A distributed DNS has been implemented as an example application that uses SRTCPU.

Table of Contents:

Uncertainty in Global Application Services with Load Sharing Policy

Mark Burgess and Sven Ingebrigt Ulland

Oslo University College, Norway
mark@iu.hio.no

Abstract. With many organizations now employing multiple data centres around the world to share global traffic load, it is important to understand the effects of geographical distribution on service quality. The Domain Name Service is an important component for global load balancing. Using controllable simulations, we show that wide area sharing can play an important role in optimization of response times when traffic levels exceed that which can be supplied by a local infrastructure. We compute the probability of being able to meet Service Level Objectives as a function of DNS caching policy (Time To Live), so that service providers can account for DNS error margins in Service Level Agreements.

1 Introduction

Meeting Service Level Objectives (SLO) is a crucial goal for online businesses, especially in the application services sector. Even if no formal Service Level Agreement (SLA) has been made, Service Level Objectives (SLO) are set by clients' demands: users will typically defect from a slow web-site after only a few seconds of waiting in order to look for an alternative[1], hence performance is directly related to profit.

One of the benefits that networking offers is the ability to use distributed resources to one's advantage. When local resources fail to cope with demand, external resources can be brought into play simply by redirecting requests to another location. That location could be a few centimetres away, or several thousand kilometres across a wide area network. However, in wide area, globalized services there are more links in the chain between server and customer where uncertainties and delays can creep in, and each of these components becomes a focus of independent interest for the performance analyst.

The role of the Domain Name Service (DNS) has been of particular interest in the matter of load sharing and redirection of traffic. On the one hand, this is an obvious place to overload existing functions for the purpose of load balancing; on the other hand it is a fragile bottleneck in Internet services with low security properties and relatively poor performance. Ten years ago, network service capacities were orders of magnitude poorer than today so the delays incurred by DNS lookups was less noticeable. Today, the DNS appears as a key bottleneck which contributes a significant fraction to service round-trip times.

In this paper, we examine the predictability of wide area load-sharing, based on the Domain Name Service, and attempt to identify strategies for minimizing its impact on Service Level Objectives.

2 Global network services

Global load sharing is a subtle topic, because it makes cost trade-offs based on quite different currencies. Many layers are involved in directing a service across the globe, and each of these can add to the uncertainty in response time, and to the delay experience by the end user. One must therefore try to unravel the various components.

Why would we not simply centralize a service for easy management? The reasons for this include security, fail-over redundancy, traffic congestion management or even power saving (or cost saving under different tariffs)[2]. The inhomogeneity of load that occurs during the course of a day often makes night-time processing favourable in a data centre[3], thus it might be a reasonable strategy to divert traffic to a geographically remote location (on the other side of the planet) in order to balance a heavy day-time load – this would assume that the routing and transport cost were acceptable[4, 5]. In other work, we have looked at how local methods can be used to predict the scaling of application services in a data centre. The ability to respond to a request depends on both the nature of demand[6] and the availability of supply[7].

There is a need then to be able to predict the performance of wide area redirections for coping with server traffic, and study them in relation to the benefits of more local strategies. We approach the question of application service performance by asking a simple question: how does the use of wide area methods for network redirection affect application services levels? i.e. What level of uncertainty does globalized load balancing add to service response times?

There is a large literature on application service modelling. Much of it is now five to ten years old, and some is contradictory. We consider the situation today.

3 DNS

The Domain Name System (DNS) is the global name lookup service in the Internet. It is a distributed, hierarchical and redundant database running on many thousands of servers world-wide, each responsible for one or more DNS zones. When a client issues a request for a URL, the browser will first try to resolve the hostname in the URL into an IP address, so that it knows where to send the request. This is where it is possible for a DNS server to influence the outcome of a query, effectively directing the client to a desired or lightly loaded site. However, the DNS approach to load balancing is not without challenges, as will be discussed shortly.

DNS servers come in a variety of flavours all of which allow the service to act as multiplexers, serving different looked-up values for incoming requests in a one-to-many mapping. The approaches differ in their 'back-ends', i.e. the amount of internal processing they use to adapt to their situation. We shall consider both static and adaptive 'back-ends' in this paper.

Consider a single DNS query, and assume that its return value has not previously been cached anywhere. The following sequence of events ensues:

1. A client program passes a partially qualified hostname to a system interface, (normally called `gethostbyname()`.
2. The operating system *qualifies* the name by adding a local domain name, if none was specified, and redirects the request to the local resolver (typically a local DNS nameserver), asking for the A-record (or AAAA-record in IPv6) for the fully-qualified name – an A-record is a standard hostname-to-IP mapping. The request to the resolver is *recursive*, which enables a flag meaning "I only want the final answer."
3. If the local resolver does not have the answer, it performs *iterative* queries through the DNS hierarchy. First it asks one of the thirteen root DNS servers. These know which servers control the top-level domains like com, org and net. The local resolver caches the response.
4. Further, the resolver asks one of the top level DNS servers for further directions to the domain. Again, it caches the response.
5. When the iteration process reaches one of the lookup domain's DNS servers, this server will know the answer to the query, and reply with an IP address, e.g. 192.0.34.166. Yet again, the resolver caches the response.
6. The response is sent back to the client operating system, which also caches the response.
7. The IP address is returned to the browser, which in turn can contact the server and retrieve the content. In addition to the operating system caching the response, many clients will do so as well.

Note how this rather simple example introduces at least five levels of caching, not counting potential intermediate http proxies.

Consider a query to a balanced web site. A typical response could be as in this example:

```
;; QUESTION SECTION:
;cnn.com.                     IN      A

;; ANSWER SECTION:
cnn.com.              300     IN      A    64.236.29.120
cnn.com.              300     IN      A    64.236.16.20
cnn.com.              300     IN      A    64.236.16.52
cnn.com.              300     IN      A    64.236.16.84
cnn.com.              300     IN      A    64.236.16.116
cnn.com.              300     IN      A    64.236.24.12
cnn.com.              300     IN      A    64.236.24.20
cnn.com.              300     IN      A    64.236.24.28

;; AUTHORITY SECTION:
cnn.com.              600     IN      NS   twdns-04.ns.aol.com.
```

The nameservers return multiple A-records for the ‘cnn.com’ hostname – this list is known as a Resource Record Set (RR set). The addresses appear to be within the same provider network. This does not mean however that they are geographically close to each other. Clients typically traverse the RR set sequentially, starting from the top. That is, if the first address on the list does not work, the client tries the next one, and so on. This is a decision that the client makes.

The numbers in the second column of the response show the Time-To-Live integer value (TTL) in seconds. This value governs how long the answer can reside in a cache in any of the intermediate nodes. Once this value has expired, the cache discards the value and the client must obtain an ‘authoritative’ answer by iterative querying once again. This is a traditional strategy for off-loading DNS servers and reduce lookup latency by caching the IP address value of a DNS lookup for the specified lifetime. The relevance of this mechanism must be reevaluated in light of performance improvements over the intervening decades.

As long as the TTL is greater than zero, queries will be answered from a cache instead of being redirected to other servers. In the example, the A-records have a TTL of 5 minutes. A low TTL ensures that DNS servers are queried often, and hence are given the possibility to obtain fresh, up-to-date information about a domain which is making changes over relatively small time intervals. Low TTLs mean frequent repetition of the arduous look-up process and hence come at the cost of adding a considerable delay to the total service response time.

For example, the DNS can consume a significant part of the time it takes to fetch a web page, which might have several in-lined objects, including pictures and advertisements from many source domains. Shaikh et al note: “25% of the name lookups (with no caching) add an overhead of more than 3 seconds for the ISP proxy log sites, and more than 650 ms for the popular sites. respectively. It is interesting to observe that nearly 15% of the popular sites required more than 5 seconds to contact the authoritative nameserver and resolve the name. This is likely to be related to the 5-second default request timeout in BIND-based resolvers”[9].

As we shall see, caching alleviates this problem considerably (indeed, the question of caching versus non-caching leads to a strongly bimodal behaviour), but with a different potential cost: global congestion.

From the above, we note that DNS can employ two basic mechanisms in multiplexing: it can cyclicly permute the RR set so that each new query is ordered differently. Since clients normally pick the first element from the list, this amounts to a continual Round-Robin shuffling of the server set. Second, the TTL value must be chosen to be a relatively low value to avoid too much re-use of old data which would bias the fair-weighted Round-Robin distribution.

Clearly, these methods are unreliable. There is much uncertainty. If we have three servers in the RR set and only every third query from a given client contains a service request, then all the traffic goes to the same server after all. We are also trusting clients to act predictably and not try to second guess the results from the DNS server by performing their own shuffling: DNS implementations are not required to preserve the order of resource record sets. As for TTL values, multiple levels of caching make it a challenge to predict and control the actual TTL observed by the end user. The TTL policy is only a polite request to caches, not an enforcable mechanism.

A problem with low TTL values is in so-called *sticky sessions* in which a user is connected to a particular instance of a network service with a cookie of session identifier that is unique to one server (e.g. in a net bank or online retailer). If the next request to a persistent session were directed to a different server, the session would be lost. We shall not discuss this particular issue here, since its resolution is a story in its own right.

4 DNS latency

Service Level Objectives are performance wishes often set informally by clients and estimated by engineers. The service provider wants to guarantee that users will gain access to their services within a Service Level Objective. Users are known to give up on slow services within just a few seconds and take their custom elsewhere[1].

The uncertainty is the response time is rightfully a combination of the uncertainty margins in each of the independent causal factors between the client and the server. This includes the identification of the appropriate server through the DNS service. One would like to write:

$$\Delta t = \sqrt{(\Delta t_{\mathrm{DNS}})^2 + (\Delta t_{\mathrm{Routing}})^2 + ...} \qquad (1)$$

Since the DNS is a network service, it is itself dependent on all of the other uncertainties in a network, so each DNS query invokes all of the latencies in the system even before a service has commenced. Alas, the inter-dependencies of a network make the Pythagorean formula above difficult to implement.

A data centre engineer would like to attempt to compensate for the lookup delay, or at least account for it in service promises to clients. Let us consider then, how much DNS server redirection could add to the margins needed for the ‘over-provision’ of services, according to this study? On a global scale there are more sources of possible delay which could add to the overall response time. It is plausible that the worst bottleneck would be shifted to some other component in the supply chain. Hence it seems possible that one might win performance improvements by making a more informed choice based upon monitoring of the available capacities along different alternative paths.

Consider the scenario in figure 1. We imagine a global organization with alternative data centres at different locations. Redirection to the appropriate data centre will occur by DNS multiplexing. There are two competing mechanisms in such a load sharing scheme: the desire to avoid bottlenecks at the dispatcher and the desire to avoid congestion at the servers themselves. If the dispatcher does not share efficiently, there might be congestion, but if the dispatcher struggles

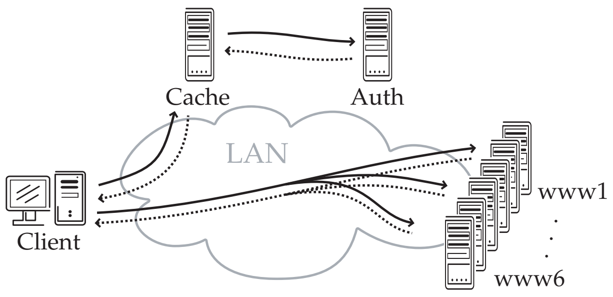

Fig. 1. A schematic illustration of the DNS load sharing scenario..

to share the load it could add to the service time itself. This is a classic problem in inventory management[10].

The DNS Time To Live is key here, since a high TTL lowers the load on the DNS as a dispatcher, but at the same time increases the congestion on the servers. It behaves as a slider-knob trading off these two effects. Figure 2 shows what one might expect for the relationship between the TTL value set in the authoritative nameserver and the round-trip time of fetching web-objects from a hostname that is registered with several IPs in the authoritative nameserver.

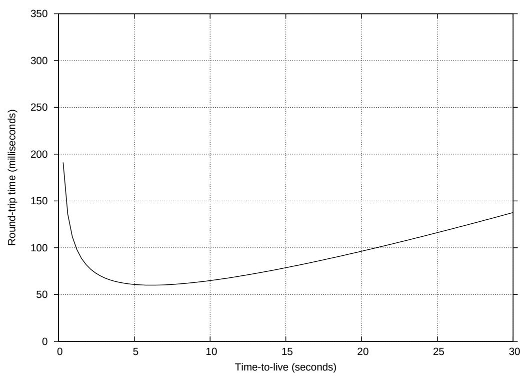

Fig. 2. Hypothesized cost of a DNS based load balancer at high traffic levels. The cost rises for low TTL as the load on the DNS server increases. The cost rises for long TTL as the individual servers start to become loaded inefficiently due to unfair weighting.

What is interesting about this form is the possibility that there exists an optimal value for the TTL parameter (the inventory re-order time). In previous work this TTL parameter has been set essentially by hand, without much insight into its functional role[8, 11]. We would like to investigate this causal role of the TTL value more carefully below by testing DNS implementations against simulated traffic patterns.

5 Empirical study

Our experiments investigate the policies that can minimize DNS induced latency. Such latency can come from the inefficiency of the DNS service itself, from the relative congestion of alternative servers, and from transport uncertainties (which are unrelated to the DNS). We arrange for the latter to disappear by isolating a DNS load balancing scenario in the lab. Hence we are left with the interplay between dispatcher (the DNS response time) and server congestion (determined by the algorithm used by the dispatcher) which yields a final service time.

We deploy a client, the client's local resolver with cache (representing a local domain nameserver), and the remote domain's authoritative DNS server. We also have six web-servers, all configured similarly to answer requests for a single hostname, e.g. www.example.net. Wide-area network emulation is provided by the NetEm facility in the 2.6.16 linux kernel[13]. This is a part of the QoS framework there, and it allows us to specify delays and delay distributions for outgoing queues to simulate load. In the experiments on TTL vs RTT, the cache has a mean delay of 20ms, the authoritative has 300ms.

5.1 Effect of TTL on round-trip time, homogeneous servers

Running the `flood` tool to test DNS server response allows us to test our hypothetical inventory processing model for the combined lookup and service time. For the initial test, we make all of the servers identical in capacity and latency.

The plot in figure 3 shows the effect of TTL on static DNS server response-time for a full page load, that is, a DNS request for the hostname, TCP connection setup, sending HTTP request and finally receiving HTTP reply (connection tear-down is not included). The HTTP request used is trivial to serve and requires few CPU cycles per request.

For TTL 0, there is little variance in the results. We found that it was essentially impossible to overload a DNS server running on modern hardware with realistic traffic intensities. The uncertainty bars corresponds exactly to a controlled delay distribution which we specified for the authoritative nameserver (using NetEm). For TTL 1 and beyond, DNS requests are served both by the authoritative and caching nameserver. Since these two servers have very different delays set up for their outward queues, a request would either be served by the cache or the authoritative server. As TTL increases, the chance for a cache hit in the caching server increases proportionally. We see a flattening out of the tail for large TTL and no apparent rise in server congestion, since the load presented by our test page was low. Thus, this data represents primarily the behaviour of the dispatcher.

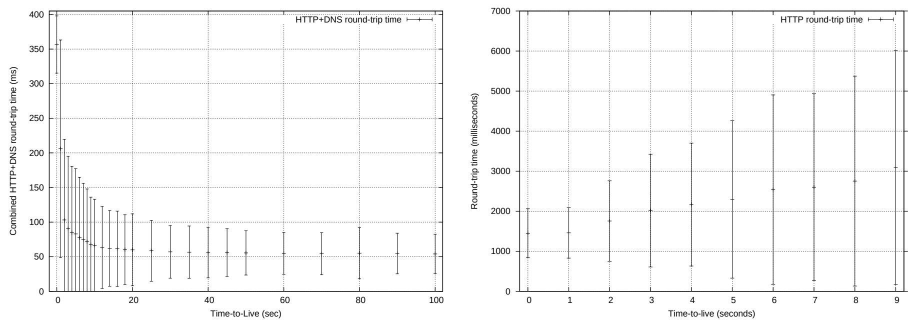

Fig. 3. Cost of lookup as a function of TTL for a 'static backend' DNS server. To be compared with the hypothesized form. Rather than rising at the end, it flattens out.. The second graph has no low TTL lookup cost up shows a rising inefficiency cost as server balancing fails..

Figure 4 shows how the DNS response time distribution is strongly bi-modal, clearly showing the influence of caching. The figure is dislocated so we can view the two peaks in more detail. We see that for TTL 0, there are no cache-hits on the caching server. But as TTL increases, so does the cache-hit rate. For TTL 100, we see a very high number of cached replies, and a close to zero amount of time-intensive authoritative requests.

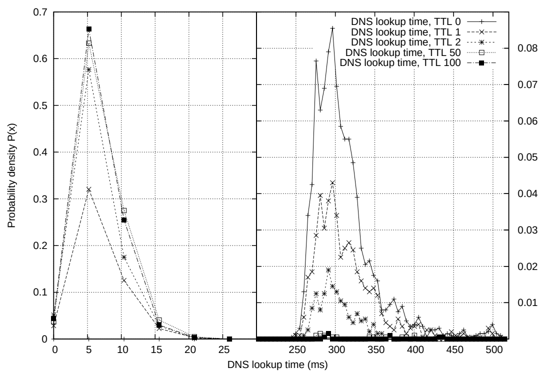

Fig. 4. The bimodal response time distribution for DNS showing the effect of caching..

The above experiment was repeated for a dynamic back-end DNS server: PowerDNS (with PostgreSQL as back-end). Here the database is loaded with the zone-data containing the resource records. PowerDNS is set up to query the database using a simple query, and appending "`ORDER BY random() LIMIT`

1", which returns one random IP address when requesting the hostname. The linearly increasing uncertainty in second figure 3 can be attributed to server loading due to poor entropy. Requests are queued and the system enters a thrashing phase. It increases with the TTL since a low TTL spreads the requests very efficiently among the servers, avoiding overload. As TTL increases, the probability of server load does too, and thus also the uncertainty. Thus we have measured both tails of our hypothetical curve for different traffic regimes.

5.2 Distribution entropy

The plot in figure 5 shows the cumulative frequency distribution of overall response times. What we observe is that for a zero TTL, most requests (90%) lie within the range of 0 to 1000 milliseconds per request. For TTL 9, we can see that approx 60% of the requests lie within the range of 0 to 2000 milliseconds per request. Also for TTL 9, we see that approx 90% of the requests lie within the range of 0 to 6000 milliseconds per request. Note that a long TTL seems to imply a longer wait, which warns against the very large TTL values suggested by authors several years ago. These results are of great interest to the Service Level Agreement architects. The efficiency of the load scheduler itself also de-

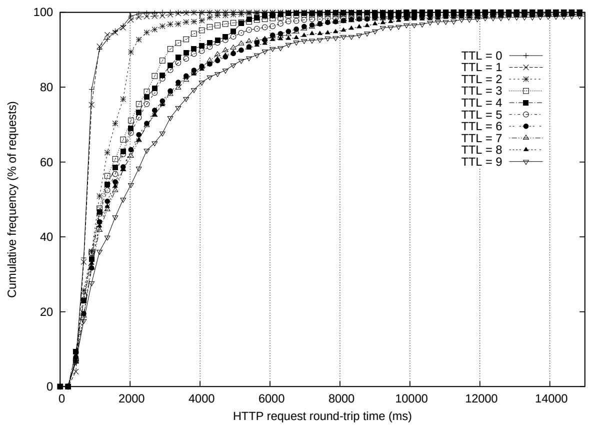

Fig. 5. Cumulative frequency plot of service times show with what certainty we can specify a response time as a function of TTL..

pends on the TTL. Ideally a load balancer will maximize the entropy of the connections histograms[14]. Figure 6 shows how well BIND distributes requests evenly among servers in a resource record set for a single client.

In the"Cyclic B" case, with a TTL of three seconds we see a more uneven response caused by BIND resetting the order of the resource record set each time the TTL expires. This causes a greater uncertainty which the simple round-robin algorithm does not cope well with: an unfair weighting is induced on the server record distribution.

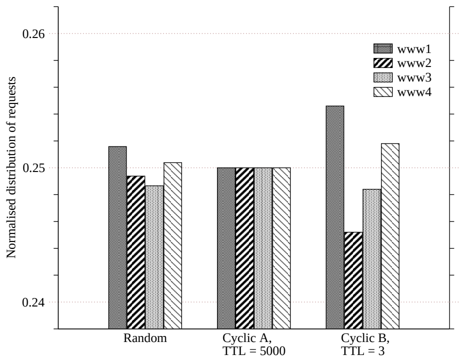

Fig. 6. Entropy of load balancing by DNS for one client..

The variations are very small, however, and do not pose any risk of over-utilisation for server 'www1' unless there is critically high traffic, in which case a non-linear instability could be seeded by this lack of parity. However, seen in the context of requests arriving from a source of many clients in diverse domains, there could be sufficient entropy of clients to even out this behaviour.

6 Comparable work

Several researchers have examined load balancing using DNS with varying conclusions. Cardellini et al survey proposed and commercially available balancing schemes, including constant and adaptive TTL schemes for DNS, dispatcher-based packet rewriting, and server-based mechanisms[8]. They find that both constant TTL with server and client state information, and the adaptive TTL scheme perform better than stateless round-robin approach.

Bryhni et al also compare a set of load-balancing implementations, with a focus on dispatcher-based systems[11]. The round-robin DNS is discussed with TTLs of 1 and 24 hours. Their trace-driven simulation results show that a dispatcher-based design running a round-robin algorithm yields the best distribution of load and amongst the lowest response times observed for that particular scenario.

In another paper Cardellini et al present a more thorough examination of web-server load balancing using DNS, and introduce HTTP redirection as a potential remedy for the otherwise coarse-grained nature of DNS[15]. They claim superior performance of redirection mechanisms over classic DNS-only balancing.

Shaikh et al, show that lowered TTL values must be carefully chosen to balance page responsiveness against excessive latency observed by the client[9].

The authors recognise that, to allow a fine-grained and responsive DNS-based server selection scheme, the TTL should be set to zero or a very low value, however this can lead to two orders of magnitude of extra delay, according to the paper. Other authors also explore the effectiveness of lowered TTL values. Jung et al [16], Teo [17], Park [18].

7 Discussion and Conclusions

DNS load balancing is a somewhat controversial topic. We have examined the behaviour of the DNS implementations with regard to their caching policy in order to find the expected uncertainty in meeting Service Level Objectives.

We find that today's DNS servers easily cope with high request volumes, in high levels (notwithstanding denial of service attacks). Caching policy does not impact directly on performance from the viewpoint of the server. However, the response time of the DNS service is relatively high by comparison to other services, due to the iterative nature of queries.

Round robin load balancing in DNS service works adequately with high levels of entropy, but are more likely to become unstable under high traffic conditions for low TTL.

Low level load balancers favour simple round-robin load-sharing at low to medium intensity[7]; there one has very low latency routes between the dispatcher and server and the cost of looking for improvements outweighs any benefits. In global routes, a DNS server can benefit from a knowledge of the round-trip time when load balancing. Unfortunately, there is not a clear correlation between the time measured by the client, and that measured by the DNS server load balancer, so this does not work well.

It is questionable whether daily load patterns can play a real role in sharing load for global companies. Response time is coupled with so many factors that it is overall response time of a dispatcher, route, and server that affects the response time the most. Of these, transport time and DNS lookup are likely the major part.

The uncertainties inherent in wide area load sharing mean that a DNS load balancing strategy is not a substitute for low level load-sharing mechanisms. Failover redundancy is a main reason for having multiple data centres, but this is not the same as load balancing by DNS. Cumulative frequency plots indicates the additional round-trip time with corresponding uncertainties for requests. This shows us what one can expect to achieve in an agreement 80% or 90% of the time. Pre-sorting sites, e.g. by 'picking the site in your country' etc preempts DNS weaknesses.

DNS cannot be avoided, but is it the right tool for load balancing? Clearly it is not. However, it is nearly the only viable *interface* for load-balancing on a global scale (IPv4 anycast is another). DNS, with the solid backing of a dynamical back-end (based on a database with state-information gathered from the servers, maybe proximity information from ARIN's IP-to-country mappings, time zone info, etc), is a very powerful tool for global server load balancing.

We are grateful to Jon Henrik Bjørnstad and Gard Undheim for helpful discussions.

References

1. J. Sauvé et al. Sla design from a business perspective. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775.*
2. M. Burgess and F. Sandnes. A promise theory approach to collaborative power reduction in a pervasive computing environment. In *Springer Lecture Notes in Computer Science*, page to appear.
3. M. Burgess, H. Haugerud, T. Reitan, and S. Straumsnes. Measuring host normality. *ACM Transactions on Computing Systems*, 20:125–160, 2001.
4. B. Abrahao et al. Self-adaptive sla-driven capacity management for internet service. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pages 557–568. IEEE Press, 2006.
5. M. Burgess. Probabilistic anomaly detection in distributed computer networks. *Science of Computer Programming*, 60(1):1–26, 2006.
6. J.H. Bjørnstad and M. Burgess. On the reliability of service level estimators in the data centre. In *Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006)*, volume submitted. Springer, 2006.
7. M. Burgess and G. Undheim. Predictable scaling behaviour in the data centre with multiple application servers. In *Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006)*, volume submitted. Springer, 2006.
8. V Cardellini and M Colajanni. Dynamic load balancing on web-server systems. *Internet Computing IEEE*, 3(4):28–39, 1999.
9. Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of dns-based server selection. In *Proc. of IEEE INFOCOM 2001*, Anchorage, AK 2001.
10. H.L. Lee and S. Nahmias. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, chapter Single Product, Single Location Models. Elsevier, 1993.
11. E Klovning H Bryhni and O Kure. A comparison of load balancing techniques for scalable web servers. 14(4):58–64, 2000.
12. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.
13. Stephen Hemminger. Network emulation with netem. In *LCA national Linux conference '05*, 2005.
14. M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems.* J. Wiley & Sons, Chichester, 2004.
15. Valeria Cardellini, Michele Colajanni, and Philip S. Yu. Geographic load balancing for scalable distributed web systems. In *MASCOTS*, pages 20–27, 2000.
16. Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. Dns performance and the effectiveness of caching. *IEEE/ACM Trans. Netw.*, 10(5):589–603, 2002.
17. YM Teo and R Ayani. Comparison of load balancing strategies on cluster-based web servers. *Transactions of the Society for Modeling and Simulation*, 2001.
18. Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *ICNP '96: Proceedings of*

the 1996 International Conference on Network Protocols (ICNP '96), page 171, Washington, DC, USA, 1996. IEEE Computer Society.

An Accounting Model for Dynamic Virtual Organizations

Matthias Göhner[1], Martin Waldburger[2], Fabian Gubler[2], Gabi Dreo Rodosek[1], Burkhard Stiller[2],

[1]University of Federal Armed Forces Munich, Information System Laboratory (IIS), D—85577 Neubiberg, Germany, goehner@informatik.unibw-muenchen.de, gabi.dreo@unibw.de
[2]University of Zurich, Department of Informatics (IFI), CH—8050 Zürich, Switzerland, [waldburger¦stiller]@ifi.unizh.ch, fgubler@gmx.ch

Abstract

The provisioning of remote and composed services in support of various application areas has dramatially increased over recent times. Thus, the concept of Grids has evolved, in the sense of a common platform for electronic service provisioning in multi-domain environments. While, traditionally, Grids have seen a quite static existence, many new service compositions have to take place on-demand and for certain periods of time only. Thus, the concept of Virtual Organizations (VO) delivers a highly suitable representation of such dynamic Grids. However, an open problem at this stage can be seen in the lack of applicable, distributed, and efficient accounting schemes for commercial resource and service consumptions. Even for management purposes, e.g., *sampling or archiving, this functionality is essential.*

Therefore, a comprehensive model for Grid accounting has been developed and suitable accountable units have been defined, in which an underlying activity- and resource-based accounting model covers economic cost theory. Furthermore, this work is based on a service model proposed for service provisioning in dynamic VOs.

1. Introduction

Grid systems have evolved over time from traditional Grid computing to service Grids. Grid computing focused on high performance computing, based on the concepts of cluster-building and resource sharing. By applying these concepts, Grid systems are able to solve data- and computationally intensive tasks. Service Grids address in addition the virtualization of resources and services across administrative borders. Thus, they dispose of mechanisms needed for geographically and organizationally distributed service provisioning within a Virtual Organization (VO).

Based on the embracing analysis performed in [31], comparing traditional Grid computing and today's service Grids with other related concepts, service Grid VOs are characterized as follows: Virtualization allows for inter-domain service provisioning and resource coordination. Resources are encapsulated by services that in turn can be aggregated into more complex service bundles. Services are accessible via well-defined interfaces and they implement standard and open protocols so that interoperability between heterogeneous Grid nodes is achieved. In contrast, implementation details as well as information about service composition and the overall organizational alignment of the VO are hidden from service users.

From an economic point of view, virtualization refers not only to the concept of resource coordination between legally independent organizational entities, it also covers dynamics with respect to organizational composition and business process execution [27]. Depending on the specific service user needs, representing a market's demand side, suited service providers need to be bound to a VO. Changes in either, demand or supply side, thus result in other operational VO instances. Furthermore, processes need to adapt to changing contexts so that *e.g.,* aggregated services have to be newly composed. In order to reflect dynamic aspects explicitly, Dynamic Virtual Organizations (DVO) are introduced and used subsequently.

Inter-domain resource coordination and service provisioning in DVOs traditionally finds most deployment cases in research-oriented Grids, but it is more and more taken up in fully competitive commercial environments as well, strengthened by the respective Grid initiatives lead by industrial supporters like IBM [24]. Either for statistical and planning reasons only, or with charging for consumed resources in mind, accounting of resource usage records is in a service provider's interest — both, in academic and commercial environments. Usage records feed accounting systems with metered statistical information on what resources have been consumed how and by which service. In economic terms, resource consumption is reflected by cost elements. An accounting system has to specify accountable units that need to follow the principles of the economic cost theory. Moreover, accountable units have to consider the specific requirements on accounting for electronic services that are offered in DVOs. This covers particularly the fact that — once a Grid service has been produced, thus, being available for service delivery — electronic services are characterized by a high share of indirect cost elements that are not directly attributable to a specific provided service.

Accordingly, the main objective of this work consists in the design of an accounting model that addresses inter-domain provisioning of electronic services in DVOs. This model needs to satisfy on the one hand DVO-specific requirements, while on the other hand, it has to consider generic requirements on accounting systems that are based on the economic cost theory. This approach is followed in order to establish an accounting system that integrates both, the perspectives of economic and technical accounting. In particular, the relevant set of accountable units for Grid services has to be specified, constituting the base components of the accounting model. In order to reach this goal a thorough investigation in existing Grid accounting concepts is performed in Section 2, including an analysis of all considered approaches. This is followed by a close inspection of DVOs and the respective service model in Section 3. The gained insights on existing approaches and Grid service characteristics lead to a set of generic and DVO-specific requirements on accountable units as determined in Section 4. Driven by these characteristics, the accounting model is developed in the same section, which is followed by a functional evaluation performed in Section 5. Finally, the work is summarized and conclusions are given in Section 6.

2. Related Work

The area of Grid accounting has already been investigated in other Grid projects and by other researchers. Therefore various approaches on accounting for Grids can currently be observed in literature. The following section provides an overview of the “status quo” of existing accounting systems and accounting tools from European as well as international Grid projects, and it finally presents an evaluation of fundamental characteristics in a condensed form.

2.1 Accounting Processor for Event Logs (APEL)

The web-based accounting tool APEL, which has been deployed within the EGEE/LCG project [18] is a log processing application used to interpret batch system and gatekeeper logs in order to generate accounting records [6]. The APEL Log Processor parses log files to extract job information and publishes it using R-GMA, an implementation of the Grid Monitoring Architecture (GMA) proposed by the Global Grid Forum (GGF) [5].

In LCG accounting, each site publishes its own accounting data using an R-GMA primary producer which makes use of its locally assigned R-GMA server. Currently, only PBS (Portable Batch System) and LSF (Load Sharing Facility) batch systems are supported by the underlying architecture. The architecture, however, can easily be extended to develop other variants. Via a secondary producer, the aggregated accounting data is streamed to a centrally administrated, relational database management system in the Grid Operations Centre (GOC), which is used to provide a web front end, generating a summary of resource usage across the Grid [6]. Furthermore, various plug-ins exists to provide access to a MySQL database in the GOC in order to publish accounting records through R-GMA for presentation on the web.

2.2 Distributed Grid Accounting System (DGAS)

The DataGrid Accounting System was originally developed within the EU DataGrid Project (EDG) [7] and has been maintained and re-engineered within the European EGEE project since April 2004 [1]. DGAS is designed to support an economy-based approach to regulating the distribution of Grid resources among authorized Grid users and to implement resource usage metering, accounting and account balancing in a distributed Grid environment [28].

In accordance with the SweGrid Accounting System (SGAS, cf. Section 2.7) all currency transactions are mediated by decentralized bank services. The consumption of Grid resources by Grid users is registered in appropriate servers, called Home Location Registers (HLR), which manage both user and resource accounts. Similar to SGAS, the Distributed Grid Accounting System makes use of so-called “template accounts” that are temporarily linked to authorized Grid users for the duration of a job submission. Furthermore, the HLR takes care of the communication between different HLRs, and it credits/debits the different users/resource owners for the respective amount of resource usage [13].

EDG makes use of an approach where each VO has their own HLR available for their members, although a finer granularity is possible. The Price Authority (PA) is an optional component in the architecture that determines prices for all different resources in the Grid, either manually or by using different dynamic pricing algorithms. Similar to the HLR approach, each VO comprises at least one PA. The costs of the user job are finally determined by the HLR service from resource prices and usage records. Account balancing is done by exchanging virtual credits (Grid Credits) between the HLRs of the user and the resource [13].

2.3 Grid Accounting Services Architecture / GridBank (GASA)

GridBank (GB), developed in Australia within the Gridbus project [23], is a secure Grid-wide accounting and payment handling system with a strong focus on economic structure and economic brokering. In [3], several requirements on Grid accounting and various economic models within GridBank are presented, based on what the Grid Accounting Services Architecture (GASA) is proposed. The paper highlights implementation issues with respect to a detailed discussion of format variants to be maintained for various records/data bases, *e.g.*, Resource Usage Records (RUR). Also several protocols for interaction between GridBank and the various components within Grid computing environments are presented. In addition, payment of resources is addressed in the approach by means of a comprehensive set of payment schemes, based on both, Grid Credits and real money.

In contrast to SGAS and DGAS, the accounting system of GridBank has a slightly different underlying infrastructure. Providers and consumers of Grid resources have to register themselves on a central server, so that resource owners do not have to establish accounts for every resource user [13]. Another interesting aspect of GridBank is that it makes use of decentralized Grid Trade Servers (GTS) to negotiate resource prices and to select an appropriate resource provider.

2.4 Grid Based Application Service Provision (GRASP)

The aim of the European GRASP project [21] consisted in integrating network-enabled Application Service Provision (ASP) with Grid computing and Web Services in compliance with the OGSI.NET framework [14]. Therefore, GRASP made use of distributed and heterogeneous resources, which are integrated using Grid technologies, in order to implement current and future ASP business models.

A fundamental concept of the GRASP infrastructure, which is compliant with the Open Grid Services Infrastructure (OGSI) specification, is the Virtual Hosting Environment (VHE). VHE comprises various services and resources, which are provided within a VO [8]. The underlying accounting subsystem, that supports usage-based and service-level-based charging of jobs and applications, is implemented by means of two basic Grid services: the Financial Accounting (FA) service as well as the charging service. The FA Service allows the execution of an application by coordinating other subsystems belonging to the Business Component Services Layer [14]. An important aspect of the accounting system is the usage of policies, which are supported by Web Service Level Agreements (WSLA) [22] as well as business extensions. The use of policies provides for accounting mechanisms that are flexible and dynamically configurable.

However, the accounting and billing services as determined within GRASP are still in an early design phase. For this reason, several interesting aspects remain unspecified, such as the range of supported resource types or accountable units.

2.5 Grid Service Accounting Extensions (GSAX)

GSAX represents an extensible OGSA (Open Grid Services Architecture) [20] accounting and logging framework as proposed by the GGF RUS-Working Group (RUS-WG) [19], and as it emerged in the course of the IBM "Extreme Blue" Grid accounting project [4].

GSAX is designed to provide a functionally modular accounting framework, which can be expanded by adding or changing so-called core components. Furthermore, the underlying accounting system allows for the use of accounting at various application levels, and it provides information at different degrees of granularity (*e.g.,* real-time information or data on a per-job basis) [4]. Another important aspect of this theoretical approach is the possibility to integrate Quality-of-Service (QoS) parameters as well as Service Level Agreements (SLA) at different levels into the accounting framework. For instance, this provides economic-based QoS and SLAs to be implemented at accounting level. The underlying accounting subcomponent comprises two basic services: first, the account management service that provides accounting-related information and accounting records to higher-level components via adequate interfaces; second, the accounting service that handles metering events and, thus, establishes interfaces with the lower components of the framework [4]. The account management service as well as the accounting service both hold an instance of an account which contains information as for example the current balance or list of users authorized to use the account.

2.6 Nimrod/G

Nimrod/G is a Grid resource management and scheduling system based on Grid technologies, which was developed at the Monash University in Australia. The tool was designed to manage the deployment of parametric experiments in a global computational Grid [2]. Nimrod/G supports an integrated computational economy in its scheduling system. This means that Nimrod/G can schedule jobs on the basis of QoS requirements, deadlines and budget restrictions [2]. The Nimrod/G Agent records the amount of resources consumed during job execution, such as CPU time and wall clock time. The online measurement of resources consumed by an executed job helps the scheduler to evaluate resource performance and to change schedules accordingly. Furthermore, information from the metering component can be used in order to perform an accounting of the resource consumption.

As a summary, Nimrod/G puts the focus on allocating and scheduling, however, not on the accounting of Grid resources. Thus, this approach lacks a specification of accountable and monetary units as well as a detailed description of user accounts and accounting records. As a result, Nimrod/G should not be considered as a single independent accounting system. Integrating Nimrod/G into existing accounting systems, *e.g.*, into GASA, proves to be reasonable [2][3].

2.7 SweGrid Accounting System (SGAS)

The SweGrid Accounting System (SGAS) is an accounting system already implemented in the Swedish National Grid (SweGrid). SweGrid is a computational Grid, initially joining together one cluster at each of six high-performance computing centers across Sweden, and currently comprising approximately 600 nodes [29].

In contrast to DGAS or GASA, the SweGrid Accounting System is built on open, standard-based Grid protocols and existing toolkits, *e.g.*, OGSI. SGAS currently supports homogeneous computing nodes as well as few accountable units only [13].

The underlying architecture of the system consists of a bank service, the Job Account Reservation Manager (JARM), and the Log and Usage Tracking Service (LUTS) [11]. In SGAS, each VO has an associated bank service in order to manage resource allocation for a given VO research project. The primary purposes of the SGAS bank are to keep track of the resource consumption for the individual projects/users and to enable coordinated quota enforcement across the SweGrid sites. Among other things, substantial features of the bank component include bank account administration, transaction history, a logging service as well as soft state account holds. JARM provides a single point of integration of SGAS into various Grid middlewares [13][15]. Moreover, when placed on each Grid resource, JARM intercepts incoming service requests, performs account reservations and charges the account of the requester after resource usage. Finally, LUTS collects and publishes on the one hand usage data, while, on the other hand, it allows users to query accounting information in a consistent way. Similar to GridBank, accounting data is stored using Resource Usage Records (RUR) as standardized by the Global Grid Forum (GGF) Working-Group [13].

2.8 Analysis of Existing Approaches for Grid Accounting

In order to conclude, the following section briefly summarizes several shortcomings of existing accounting approaches for Grid environments. A detailed summarization of fundamental characteristics of existing accounting systems is depicted in Figure 1.

✓ „Yes“, (✓) „In parts“, – „No“, n. s. „Not specified“.

Criteria	Accounting System						
	APEL	DGAS	GASA	GRASP	GSAX	Nimrod/G	SGAS
Interoperability und portability	(✓)	(✓)	(✓)	n. s.	(✓)	✓	✓
Scalability	✓	(✓)	–	n. s.	✓	✓	✓
Integration	(✓)	(✓)	(✓)	n. s.	(✓)	✓	✓
Inter-organizational accounting	✓	✓	✓	n. s.	✓	n. s.	✓
Flexibility und extensibility	✓	n. s.	✓	n. s.	✓	(✓)	✓
Support of existing standards	–	–	(✓)	(✓)	(✓)	n. s.	✓
Support of multi-provider scenario	–	–	–	–	–	–	–
Customer-specific visualisation of accounting data	✓	–	–	n. s.	n. s.	n. s.	–
User transparency	n. s.	n. s.	n. s.	n. s.	n. s.	n. s.	(✓)
Accounting of heterogeneous resources	(✓)	✓	✓	(✓)	n. s.	n. s.	–
Accounting of virtual resources	–	–	–	–	–	–	–
Accounting of virtual services	–	–	–	–	–	–	–
Virtualization concept	–	–	–	–	–	–	–
Support of high dynamics	✓	(✓)	(✓)	n. s.	n. s.	✓	✓
Security	n. s.	✓	✓	n. s.	✓	n. s.	✓
Standardized, generic interfaces	–	–	–	n. s.	(✓)	✓	(✓)
Support of various accounable units/metrics	✓	✓	✓	n. s.	✓	n. s.	–
Precision and abundance	✓	✓	✓	✓	✓	n. s.	✓
Support of different accounting policies	✓	✓	n. s.	n. s.	✓	n. s.	(✓)
Reliability and fault tolerance	n. s.	n. s.	(✓)	n. s.	n. s.	n. s.	✓
Administration and management	n. s.	(✓)	n. s.	n. s.	n. s.	n. s.	✓
Verification	n. s.	✓	✓	n. s.	n. s.	✓	✓
Open Source	✓	✓	✓	–	–	✓	✓

Figure 1: Evaluation of Existing Approaches

One of the major drawbacks of existing accounting systems is that they do not offer a comprehensive concept for the virtualization of resources and services with respect to accounting. Neither of the considered accounting approaches defines virtual resources or virtual services, nor do they provide mechanisms for the accounting of virtual resources and virtual services as they are offered within VOs. Generally, the focus of existing systems and accounting tools is on the accounting of physical Grid resources. Accounting of complex services — as for instance information services or computation services — is considered only partially. In addition, the aspect of accounting of composed virtual resources and services is also not reflected: this is of particular importance for multi-provider scenarios where several real and virtual organizations provide together a virtual resource or virtual service. Furthermore, to some extent only static environments with Grid resources of homogeneous nature are considered by the underlying accounting systems. Dynamic Grid environments with a high level of heterogeneity with respect to resources, operating systems, and Grid middleware are in most cases not taken into consideration.

3. Service Model for Dynamic Virtual Organizations

Due to lacking support of virtualization concepts and multi-provider scenarios in existing Grid accounting approaches, a closer look at characteristics of electronic service provisioning in DVOs is needed. To this purpose, a generic service model for DVOs needs to be determined. This section's main objective, thus, is to refine the DVO model introduced in [10] as well as the proposed model in [30] in order to specify a comprehensive service model for Dynamic Virtual Organizations, as Figure 2 illustrates.

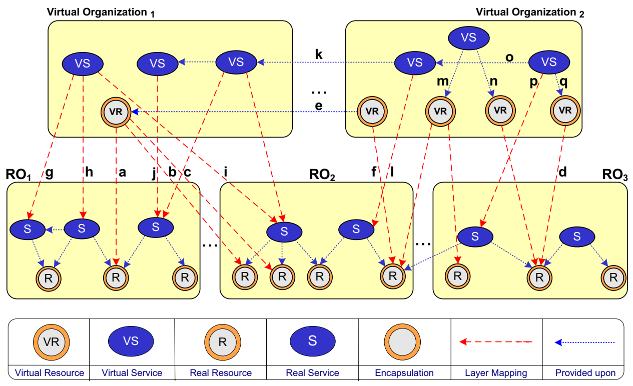

Figure 2: Service Model for Dynamic Virtual Organizations, extended from [30]

In the process of creating DVOs, multiple autonomous organizations — so-called real organizations (RO) — are involved in providing the VO's resources and services by contributing some of their local resources and services, respectively. DVOs are characterized by strong dynamics in their organizational composition and business processes. Furthermore, in a highly dynamic VO, resources are not necessarily dedicated to a single service or a single organization only. In the context of Grid computing, virtualization can be seen as the mapping of real objects (such as resources, services and organizations) into virtual objects, which may have functional characteristics of (different) real objects [32]. In Grid environments, the concept of virtualization provides an essential means to increase the flexibility within VOs with respect to, *e.g.*, the provisioning of resources as well as services. If, from a service-oriented point of view, the focus of VOs is on the provisioning of services, a VO can be defined as a set of virtual resources and virtual services that can be used by individuals to archive a common goal [10]. Furthermore, resources and services of a VO may also be provided to members of other VOs. This case visualizes exemplarily how important it is for an accounting system to account for resource consumption and service usage in an accurate form.

A virtual resource of a VO may consist of several physical resources provided by one or multiple real organizations. A fundamental characteristic of Grid resource virtualization is that they may represent resources — or rather types of resources — that do not physically exist in this way within real organizations. A VO, for example, may provide a virtual storage resource which consists of several storage elements from real organizations, *e.g.*, a RAID system of RO_1 (cf. line a in Figure 2) as well as a file system and a tape storage system of RO_2 (lines b and c). Altogether, these resources build a virtual storage resource offered by VO_1.

Normally, the functional properties of a virtual resource are assumed to be the same as the ones of the real resource, but the non-functional characteristics may be different [32]. In the simplest case, a virtual resource offered within a given VO consists of exactly one resource of an underlying RO (line d). In addition, virtual resources may also comprise a set of virtual and real resources. For instance, VO_2 could provide a virtual resource that consists of a virtual computing resource (line e) of VO_1 which in turn contains several computing clusters as well as a physically existing resource of RO_2 (line f), *e.g.*, a high-performance computer. Finally, the concept of virtualization can also be applied to virtual resources itself so that a virtual resource may consist of several other virtual resources of potentially different VOs in order to provide a compound virtual resource.

In Grid environments, physical Grid resources, such as computing elements, storage resources, networks, software licenses or scientific devices may exhibit a high level of heterogeneity. Therefore, resource virtualization is also used to provide a homogenized view on heterogeneous resources. For that reason, resource virtualization can be seen an essential means to reduce the complexity of managing heterogeneous systems and to handle diverse resources in a uniform way [12].

In order to provide the functionality of a resource, as for example access to a database or computation elements, real as well as virtual resources are encapsulated by the use of a trivial service which provides standardized access to the resources in an abstract way. From a technical perspective, these encapsulation services for example may be realized as Web Services. Beside resources, a VO can also comprise a set of virtual services that are composed of several services of one or more real organizations participating in the VO (lines g, h, and i). These so-called compound virtual services [10] are perceived as instantiations of multiple, potentially different services from real organizations onto one virtual service. The underlying services of the ROs — which themselves are provided upon physical resources — can be of homogeneous or heterogeneous nature. Similar to the provision of virtual resources, a VO may also offer virtual services, which use several real services in order to build up a new type of service having a new functionality and which is not provided as such within the underlying organizations.

The following "virtual information service" sketches an example how a compound virtual service comprising several services from the underlying real organizations might look alike: The virtual information service collects information on experiments in the domain of particle physics, using a data service of RO_1. It then performs several calculations on this information via a computation service, offered by RO_2, and finally presents the results in graphical form using a visualization service offered within RO_3. In this example, the virtual information service makes use of several underlying services from real organizations which are provided upon physical resources that altogether build a composed virtual service. Through virtualization, a virtual service in most cases becomes a more complex service than the underlying services of the real organizations [10]. In the simplest case, a virtual service may consist of exactly one service from the underlying RO (line j). In this case, one-to-one mapping from the real service onto the virtual service takes place. Furthermore, a virtual service may also comprise several real and/or virtual services in order to build a new compound service. For example, a virtual service of VO_2 may use another virtual service from VO_1 together with a real service from RO_2 in order to provide a new compound virtual service (lines k and l).

On the other hand, in analogy to service provisioning in real organizations, a VO may also offer virtual services that are not composed of virtual and/or real services, but which are provided upon virtual resources (lines m and n). An example for this case would be a virtual computation service offered by VO_2 that is provided upon virtual computing resources within VO_2, which in turn are composed of several computing facilities of the underlying organizations. Finally, a virtual service can make use of real and/or virtual services and additionally may consume one or several virtual resources in order to offer a new virtual service (lines o, p, and q).

Generally, the set of virtual resources and virtual services needs to be mapped in an operational manner to real resources and real services by applying an adequate mapping function [10]. The mapping function is also an important means to map accountable units for virtual resources/services onto real resources/services. In order to access services or resources of another VO, it is necessary to have an established trust relationship between VOs, which in turn is carried forward to all services and resources of the underlying ROs. VO members only have immediate access to virtual services and virtual resources of another VO. Thus, they can not directly use resources and services of the underlying organizations which are arranged in the lower layer of the service model.

4. Accountable Units for Grid Services

Driven by the comprehensive analysis of existing Grid accounting approaches (cf. Section 2), and the presented service model for DVOs, the relevant requirements on accounting mechanisms in DVOs are identified in this section. This includes both, generic and DVO-specific requirements. In a second step, the determined requirements are considered in developing an embracing accounting model for DVOs.

4.1 Requirements on Accountable Units

In general, accounting systems rely on accountable units. These determine the range of possibilities for taking records of resources consumed during provision of a service or product. Resource usage records thus form the basic input for accountable units that, in turn, serve as input for charge calculation. Accountable units depend on the respective range of measurable units of a specific resource. Resources may be both, tangible and intangible, *e.g.,* a piece of hardware or software.

Accountable units, in their role of the basic constituting elements of an accounting model, have to satisfy a set of generic and application environment-driven requirements. The first category embraces generic accounting practices, while the latter covers in this context DVO-driven requirements, well considering the applicable presented service model for DVOs. With respect to generic accounting requirements, the following issues are of particular concern:

Typically, accounting is divided into internal/managerial and external accounting. This separation is reasoned by the different objectives the variants take. While external accounting (also referred to as financial accounting) is bound to informing organization-external entities, such as investors or authorities, internal accounting serves mainly business-optimizing purposes. Accordingly, external accounting is highly regulated, whereas organizations are free on what accounting approach they follow for internal aims. Internal or managerial accounting is also called cost accounting, since costs are perceived as an important element on the one hand to estimate process efficiency. On the other hand incurred costs also serve as input to price calculation. Driven by the main aim of this work — consisting in the design of an accounting model for DVOs — accounting is understood as *internal* or *cost accounting* only. In the context of a DVO, which deals with resource coordination across administrative borders, cost accounting is particularly relevant, since costs, by definition, directly express resource consumption.

In accounting, causality between given facts to be conceived by an accounting system and assigned accountable units is essential. Transferring this concept to cost accounting means that for every cost element the corresponding cost driver needs to be determined. Cost drivers, hence, stand for the event or fact that primarily has caused costs in the first place.

Beside those generic requirements, application domain-specific requirements need to be considered. In a DVO, primarily electronic products – in terms of electronic Grid services – are offered. With the focus on electronic service provisioning often comes a high relative amount of indirect costs (overhead costs). While labor and material costs are relatively easy to be assigned directly to products, it is more difficult to allocate indirect costs to products. Where products use common resources differently, a weighting mechanism is needed for the cost allocation process. This assumption is valid for any electronic service that is not production-oriented, *i.e.*, once a given service has been produced for the first time, re-provision of the same or similar service does not cause costs that grow linearly with the number the service has been delivered. In such a case, the share of costs that are directly assignable to the provision of a service diminishes, while the relative amount of indirect costs (*e.g.,* depreciation on infrastructure) increases with the overall number of service deliveries. As a consequence, for DVOs which face a high amount of indirect costs to be assigned, the chosen accounting model needs to provide methods that support allocation of indirect costs.

Another DVO-specific requirement is derived from the presented applicable service model that puts focus on virtual services and resources in a multi-provider environment. Allowing basic services to be aggregated into more complex service bundles demands for an accounting model that provides the means to generate different views of aggregation on accounted data. In doing so, the consistent use of the same accounting practices on all aggregation levels is desired. Hence, the accounting model is required to be highly flexible and universally applicable, so that it can reflect, in the same way, the respective views of a single Grid service provider, or of a VO as a whole.

If all those generic and DVO-specific requirements on accountable units are taken into account, the following overall claim is determined: Accountable units have to be specified in the most flexible way, since they form the base building blocks every Grid service can be composed of. These accountable units have to act as the cost objects on which indirect cost elements can be assigned according to suited cost drivers. By means of the set of specified accountable units, the accounting model needs to bridge the gap between financial data, originating from the traditional cost accounting systems, and the technical accounting system.

4.2 Development of an Accounting Model for Dynamic Virtual Organizations

Section 2.8 reveals that the latest Grid researches primarily focus on the accountability of Grid services from a technical perspective and on a meta level of VOs. All presented Grid accounting models and architectures deal with resource usage records or offer a usage tracking service. In order to use Grid services in a commercial environment, economic and financial principles have to be considered and actual costs have to be allocated to the resource usage of a service. Furthermore, DVOs are characterized by virtualization and dynamics which makes service provisioning very complex as managing heterogeneous systems and diverse resources from different service domains need to be considered. These facts demand for the definition of accountable units and an accounting systematic that satisfies the above mentioned characteristics and that fills the identified gap of already existing accounting models. The accountable units presented in this paper form a link between the financial world and the technical Grid environment.

Although Grid services are very heterogeneous and rely on the usage of different resources, accountable units need to be determined from the consistent set of base building elements out of which every Grid service is composed. This approach facilitates both, flexible and consistent accounting and charging. These universal accountable elements are called service constituent parts, covering namely Processing, Storage, Transferring and Output, as Figure 3 shows. These four service constituent parts represent the four basic hardware functionalities, out of which any intangible digital service is assembled by some amount. This does not imply that all four service constituent part types have to be used for generating a given service. In order to allow a better understanding of a service constituent part, the four service constituent part types are explained in Table 1 with respect to detailed descriptions, cost elements, applicable metrics, and relevant cost drivers.

After the introduction of the four basic building components, an accounting model is going to be introduced. The model considers the requirement to support virtual multi-provider scenarios. The main target of any participant in the DVO is to know the costs for providing one specific service request. Figure 3 provides an overview of the idea of the accounting system and the central role of service constituent parts. The accounting model relies on two well-known accounting systems.

The first is the Traditional Cost Accounting System (TCAS) as specified in supposably every cost accounting

Table 1: Service Constituent Part Characteristics

Constituent	Characteristics (Description, Costs, Metrics, and Cost Drivers)
Processing	It is very unlikely that Grid service provisioning without data processing is possible at all. As a result, the service constituent part "Processing" receives strong importance. Basically, this service constituent part type calculates the cost for processing data in a CPU. The IT resource for this service constituent part type, thus, is considered to be a CPU. Processing costs primarily depend on the used hardware (CPU Type) and also the used application logic (software). In addition to that, other costs like cooling devices for CPUs as well as cache memory could be considered too. The activity metric for "Processing" is based on CPU usage which can be either measured as CPU seconds or million instructions per second. Therefore, the service cost driver for "Processing" is the time of CPU usage (CPU cycles).
Storage	This service constituent part type calculates costs for storing data during a certain amount of time. The main cost-causing resources for this service constituent part type are data storage devices, such as hard disks or any portable data carriers. The costs for this service constituent part type also depend on the costs of all hardware components around the storage system, such as racks, software for managing the hard drives as well as back-up systems, magnetic tapes, and robot systems for exchanging and transportation. Possible activity metrics for this service constituent part type are: I/O operations, transmitted volume, used disk or tape space over a certain time period. The more data has to be stored, and the longer it has to be saved, the more costs are incurred. In consequence, a suitable service cost driver for "Storage" is the mathematic product of volume and time.
Transferring	This service constituent part type calculates costs on the one hand of transferring data between resources within a real organization (internal transferring). On the other hand, costs for external data transfers between virtual resources are considered, *e.g.,* between RO and VO or VO and VO. Basically, these transferring costs can be separated in WAN costs and LAN costs, since both categories typically show different cost structures. Moreover, external and internal data transfer might differ in the way of what Quality-of-Service levels can be guaranteed. The IT resource consumed by this service constituent part type is a "Network" of a specific administrative domain. Service cost drivers for "Transferring" could be the transferred data volume or the provided bandwidth.
Output	This service constituent part type calculates costs for generating a tangible output. Possibly, this could be a printed document, *e.g.*, an invoice for charging purposes, forms, or photos. For example, this service constituent part type is provided by a printing centre in a bank, generating an overview of the monthly transactions of a bank account. The cost-causing IT resource for this service constituent part depends on the hardware (*e.g.*, printers) used. Apparently, printing is the only service constituent part that uses directly clearable material consumption. From a cost-effectiveness perspective, this service constituent part type is characterized by a relatively high share of direct cost elements and variable cost when compared with the other service constituent part types where indirect overhead costs prevail. Thus, the more pages printed, the more toner, ink and paper will be used, and the higher the costs will be. Cost-influencing factors are output device-dependent characteristics (ink printer or laser printer), time (priority), print quality, and paper quality. An adequate metric for measuring the consumption of this service constituent part is printed pages or printed lines. Instead of printing a document, it is also possible that this service constituent part describes the process of storing data on a portable data carrier such as recordable CDs or magnetic tapes. Accordingly, other service cost drivers have to be defined (*e.g.*, MB burned, used CDs or DVDs).

text book, such as in [16]. TCAS is used to allocate costs first from financial data to cost centers and then to cost objects which are the corresponding service constituent parts. TCAS is therefore used to determine cost rates for the service constituent parts involved in providing a specific service.

The second accounting system is Activity Based Costing (ABC) [26][25][17]. The main idea of ABC is that cost objects consume activities. Activities, in turn, consume resources, and the consumption of resources determines the event that drives cost. Instead of using broad, arbitrary percentages to allocate costs, ABC seeks to identify cause-and-effect relationships in order to assign costs objectively. Once costs of the activities (represented by service constituent parts) have been identified, the costs of every activity are attributed to each product to the extent that the product uses the activity. In this way, ABC often identifies areas of high overhead costs per unit and, thus, directs attention to finding ways to reduce costs or to charge more for costly products. A big advantage of ABC is its hierarchical and modular structure which is useful for accounting at various levels of application and granularity. This allows for universal applicability. A more detailed illustration of ABC accounting is provided by Figure 4.

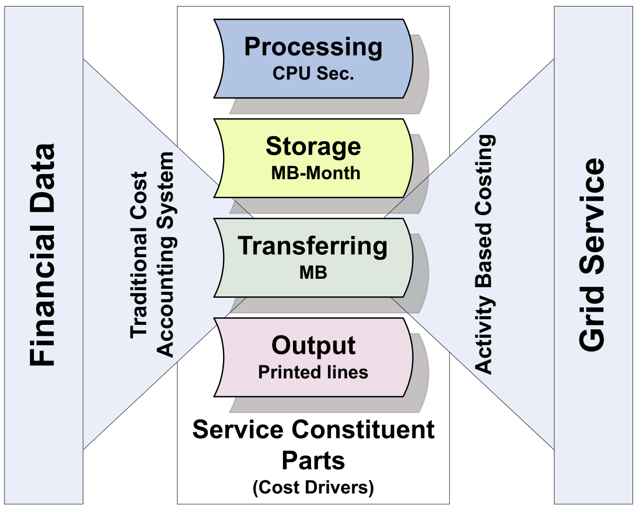

Figure 3: Accountable Units Overview

The most difficult task, however, consists in finding the appropriate amount of service constituent parts. The higher number of service constituent parts to be considered, the more accurate costs will be collected, but the more expensive the measurements will be. Therefore, careful considerations about economic cost/benefit calculation have to be made. There is no generic rule for choosing the right amount of service constituent parts so far — RO may use a bottom-up approach while VO may use a top-down approach. The bottom-up approach identifies first every service constituent part offered by an IT resource, while the top-down approach starts with the offered service and identifies the real and virtual resources (from different VOs or ROs) with their corresponding service constituent parts used for service provisioning.

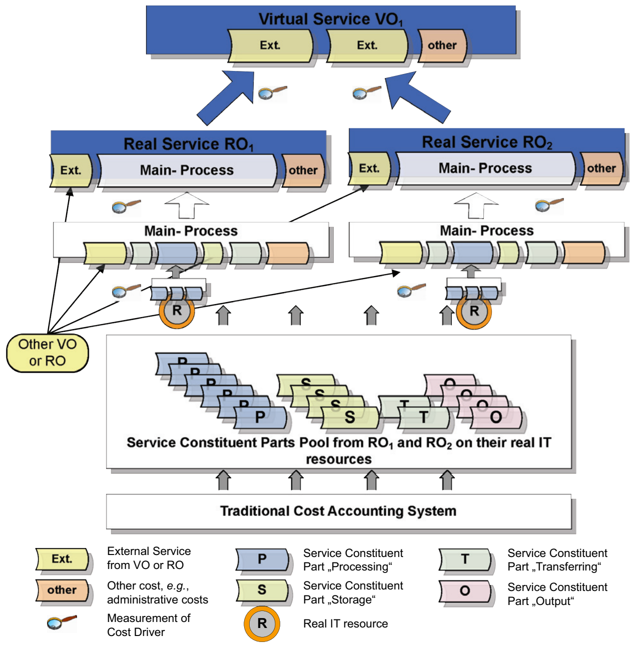
Virtual Service VO1
Ext.
Ext.
other
Real Service RO1
Ext.
Main- Process
other
Real Service RO2
Ext.
Main- Process
other
Main- Process
Main- Process
R
R
Other VO or RO
P
S
T
O
Service Constituent Parts Pool from RO1 and RO2 on their real IT resources
Traditional Cost Accounting System
External Service from VO or RO
Other cost, e.g., administrative costs
Measurement of Cost Driver
Service Constituent Part „Processing“
Service Constituent Part „Storage“
Real IT resource
Service Constituent Part „Transferring“
Service Constituent Part „Output“

Figure 4: ABC Accounting Model for a DVO

As Figure 4 shows, the usage of ABC accounting allows to allocate also other costs for service provisioning which are not chargeable to any of the above mentioned service constituent parts. This could be organization-specific cost elements such as, *e.g.*, administrative costs that accrue due to service provisioning. From the perspective of a VO, external services (virtual services) could be charged in a flat fee scheme or per service request. In the example given in Figure 4, VO_1 offers a virtual service that is composed by two external services, the first provided by RO_1 and the second by RO_2. In addition to the costs incurred by sourcing those external services, additional costs are included on VO level, for instance for administrative activities. Focusing now on the first external service provided by RO_1, the example shows what cost-relevant activities are needed for RO_1 to provide this service to VO_1. Again, an external service is sourced from a third party, followed by RO_1's main process, plus other, not further specified, cost elements. Within the administrative domain of RO_1, several information aggregating steps are taken, leading in a top-down approach to a fine-granular process cost analysis, until, on the lowest level, the respective service constituent part assignment per real IT resource is conducted. Even though the example depicted in Figure 4 features real resources only, the accounting model is applicable in the same way to virtual resources and virtual services (cf. Section 3) as well.

5. Evaluation of the Accounting Model

The objective of the following chapter is to provide an evaluation of the accounting model with regard to the requirements determined in Section 4.1 and to finally point out benefits as well as shortcomings of the presented accounting systematics.

From a theoretical viewpoint, the proposed accounting model fully matches the set of generic as well as DVO-specific requirements on accountable units. The presented approach is compliant with the service model introduced in Section 3, and it covers all relevant aspects of service provisioning in DVOs from an accounting perspective. Furthermore, the accounting model provides the possibility to bridge concepts of the Traditional Cost Accounting System and technical accounting, allowing to allocate financial expenses to resources and to calculate costs for resource usage by means of cost drivers. In addition, the proposed accounting model is also highly flexible as it is able to reflect different views within the same model *e.g.,* the view of a single Grid service provider and multiple Grid service providers as well as the view of a VO as a whole. Thus the accounting model is universally applicable in highly dynamic environments being capable to reflect the perception of different Grid service providers. Moreover, the accounting model for DVOs is extensible in the sense that, on the one hand it can be adapted and expanded concerning the objectives of different Grid service providers. On the other hand, costs for additional resources – as for example scientific devices – may also be taken into consideration by the accounting model by the use of the service constituent part „other“. The underlying idea of allocating costs first to IT resources and then to the four service constituent parts additionally makes the accounting model very powerful for an application in dynamic environments. Due to the fact that within the presented approach basically only fixed overhead costs are allocated to the service constituent parts, the accounting model allows for the detection of inefficiencies whereas an economical analysis is simply done by identifying IT resources which do not run at a high workload level.

Since accounting on a fine-granular basis causes a considerable effort which stands in contrast to the overall benefit of the accounting system, a cost/benefit analysis still has to be performed. Within this work, the requirements match has been performed from a theoretical viewpoint. The accounting model however cannot be evaluated in practical terms. Thus, its application in a real-world scenario has to be envisaged, as for instance within the German D-Grid [9] where several so-called Community-Grids (*e.g.,* MediGrid, AstroGrid, HEP-Grid, etc.) and resource providers (*e.g.,* supercomputing centres, universities, etc.) jointly offer a broad range of complex Grid resources and Grid services to the German scientific community.

6. Summary and Conclusions

The emerging importance of service Grids and its associated concept of DVOs, to allow a flexible and dynamic grouping of virtual services and resources, has brought up also the importance of accounting in such a highly dynamic environment. A comprehensive investigation into accounting models for Grid systems has revealed that existing approaches do not consider the full set of requirements on Grid accounting in DVOs, covering in particular aspects of resource and service virtualization as well as accounting in multi-provider scenarios. This has led to the need for an integrative Grid accounting model to be determined. Since DVOs provide virtual services with certain service levels and costs, the identification of accountable units and accounting models is essential as well. In the paper an activity based costing, resp. accounting model for DVOs is proposed and evaluated. Besides, accountable units have been identified, i.e., four basic service constituent parts of processing, storage, transferring and output. Due to the fact that the model was designed following economic accounting principles, rather than feature-driven requirements only, it qualifies as the first Grid accounting model to bridge the apparent gap between financial and technical accounting systems. From a conceptual point of view, the developed accounting model is sustainable. Results, however, are still of a theoretical nature at this point. Therefore, it is planned to verify its bearing strength in a real-word scenario in the area of D-Grid, an initiative to establish e-science in Germany.

Acknowledgements

This paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

[1] C. Anglano, S. Barale, L. Gaido, A. Guarise, G. Patania, R. Piro, F. Rosso, A. Werbrouk: *The Distributed Grid Accounting System (DGAS)*; (Accessed) June 2006. http://www.to.infn.it/grid/accounting/main.html.

[2] A. Barmouta*: Authorization and Accounting Services for the World Wide Grid*; Master thesis, University of Western Australia, June 2004, pp. 1-154.

[3] A. Barmouta, R. Buyya: *GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration*; 17th Annual International Parallel and Distributed Processing Symposium (IPDPS 2003), April 2003, pp. 22-26.

[4] A. Beardsmore, K. Hartley, S. Hawkins, S. Laws, J. Magowan, A. Twigg: *GSAX Grid Service Accounting Extensions*; September 2002, pp. 1-30. http://www.doc.ic.ac.uk/~sjn5/GGF/ggf-rus-gsax-01.pdf.

[5] Byrom, R., R. Cordenonsi, L. Cornwall, M.Craig, D. Abdeslem, A. Ducan, S. Fisher, J. Gordon, S. Hicks, D. Kant, J. Leake, R. Middleton, M. Thorpe, J. Walk, A.Wilson: *APEL: An Implementation of Grid Accounting Using R-GMA*; September 2005, pp. 1-2. http://www.gridpp.ac.uk/abstracts/allhands2005/apel.pdf.

[6] Byrom, R., J. Walk, and D.Kant: *User Guide for APEL - Accounting Using PBS Event Logging*; March 2005, pp. 1-32. http://hepunx.rl.ac.uk/edg/wp3/documentation/apel/apel-user-guide.pdf.

[7] The DataGrid Project: *The DataGrid Project*; (Accessed) June 2006. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[8] D-Grid Initiative: *D-Grid AK2 Middleware und Services Bestandsaufnahme*; Mai 2004, pp. 1-8. http://grid.desy.de/d-grid/ak2/DGrid_AK2_Ergebnisbericht.pdf.

[9] D-Grid Initiative: D-Grid Initiative; June 2006. http://www.d-grid.de/.

[10] G. Dreo Rodosek, H.-G. Hegering, B. Stiller: *Dynamic Virtual Organizations as Enablers for Managed Invisible Grids*; 2006 IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), April 2006, pp. 1-11. ftp://ftp.ifi.unizh.ch/pub/techreports/TR-2005/ifi-2005.09.pdf.

[11] E. Elmroth, P. Gardfjäll, O. Mulmo, A. Sandgren, T. Sandholm: *An OGSA-Based Bank Service for Grid Accounting Systems*; 2nd International Conference on Service Oriented Computing, June 2004, pp. 279-288.

[12] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich: *The Open Grid Services Architecture*; Version 1.0: Informational Document, Global Grid Forum (GGF), January 2005, pp. 1-61. http://www.ggf.org/documents/Drafts/draft-ggf-ogsa-spec.pdf.

[13] C. Frogner, T. Mandt, S. Wethal: *Cluster and Grid Computing: Accounting and Banking Systems*; Study Project Report, May 2004, pp. 1-59. http://hovedprosjekter.hig.no/v2004/data/gruppe05/files/cgc2004_report.pdf.

[14] Gagliardi, F., P. Graham, M. Heikkurinen, J. Nabrzynski, A. Oleksiak, M. Parsons, H. Stockinger, K. Stockinger, M. Stroinski, J. Weglarz: *GRIDSTART Project: IST Grid Projects Inventory and Roadmap*; March 2005, pp. 1-281. http://www.gridstart.org/download/GRIDSTART-IR-D2.2.1.4-V1.6.doc.

[15] P. Gardfjäll: *SweGrid Accounting System Bank*; December 2003, pp. 1-14. http://www.sgas.se/docs/SGAS-BANK-DD-0.1.pdf.

[16] R. H. Garrison, E. Noreen, P. C. Brewer: *Managerial Accounting*; 11th Edition, McGraw-Hill/Irwin, December 2004.

[17] J. Gerlach, B. Neumann, E. Moldauer, M. Argo, D. Frisby: *Determining the Cost of IT Services*; Communications of the ACM, Vol. 45, Nr. 9, September 2002, pp. 61-67.

[18] J. Gordon: *Accounting, 'the last A'*; Workshop Presentation, May 2005, pp. 1-18. http://www.e-irg.org/meetings/2005-NL/johngordon-accounting.pdf.

[19] The Global Grid Forum: *OGSA Resource Usage Service WG (RUS-WG)*; (Accessed) June 2006. https://forge.gridforum.org/projects/rus-wg.

[20] The Globus Alliance: *Globus: OGSA - The Open Grid Services Architecture*; (Accessed) June 2006. http://www.globus.org/ogsa/.

[21] The Grid-based Application Service Provision (GRASP) Project: *GRASP*; (Accessed) June 2006. http://eu-grasp.net/english/default.htm.

[22] The Grid-based Application Service Provision (GRASP) Project: *GRASP Tutorial, First Presentation: SLA Document, Manageability and Accounting Subsystem*; February 2005. http://eu-grasp.net/english/SalernoMeeting/GRASP%20Tutorial%20Final%20-%20Verdino.ppt.

[23] The Gridbus Project: *The GRIDS Lab and the Gridbus Project*; (Accessed) June 2006. http://www.gridbus.org.

[24] IBM: *IBM Grid Computing*; September 2005. http://www-1.ibm.com/grid/.

[25] R. S. Kaplan, W. Bruns: *Accounting and Management: A Field Study Perspective*; Harvard Business School Press, December 1987.

[26] R. S. Kaplan, A. A. Atkionson: *Advanced Management Accounting;* Third Edition, Prentice Hall, January 1998.

[27] B. R. Katzy: *Design and Implementation of Virtual Organizations*; Thirty-First Hawaii International Conference on System Science, Vol. 4, January 1998, pp. 142-151.

[28] R. Piro, A. Guarise, and A. Werbrouk: *An Economy-based Accounting Infrastructure for the DataGrid*; Fourth International Workshop on Grid Computing, November 2003, pp. 202-204.

[29] T. Sandholm, P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo: *An OGSA-Based Accounting System for Allocation Enforcement Across HPC Centers*; 2nd International Converenc on Service Oriented Computing, November 2004, pp. 279-288.

[30] M. Schiffers: *Modelling Virtual Organizations in Grids*; Technical Report, Department of Informatics, Ludwig-Maximilian University, Munich, Germany, 2006, to appear.

[31] M. Waldburger, B. Stiller: *Toward the Mobile Grid: Service Provisioning in a Mobile Dynamic Virtual Organization*; University of Zurich, IFI Technical Report 2005.07, August 2005, pp. 1-12. ftp://ftp.ifi.unizh.ch/pub/techreports/TR-2005/ifi-2005.07.pdf.

[32] T. Weishäupl: *Business and the Grid - Economic and Transparent Utilization of Virtual Resources*; PhD thesis, University of Vienna, December 2005, pp. 1-133. http://www.pri.univie.ac.at/~weishaeupl/tw-thesis.pdf.

System Administration and the Business Process

M. Burgess

November 20, 2005

1 Introduction

The integration of computer technology into a business process is an obvious goal for an organization or company that relies on information services. In this chapter we ask: is it possible to understand business modelling and analytical system administration in the same light? In other words, how do we bring a rational scientific method to bear on the problem of building a business around an Information Technology (IT) infrastructure, or building an IT infrastructure for an existing business?

To do this, we must view process of business modelling in the same light as the process of modelling the rest of the information system, and to relate these ideas to the practical issues that businesses have to contend with.

The human component of both businesses and human-computer systems requires us to consider both rational decision making (optimization based on objective criteria) and irrational choices (arbitrary choices made as policy for reasons that are not part of a scientific framework, e.g. ethics or image). For example:

- Standards of “best practice”. Why are they best?
- Modelling a business process as an information system.
- Logistics of production and inventory.
- Identification of relevant scales of operation.
- Capacity planning.
- Time management.
- Strategic thinking: activity patterns like type I models, and decision planning like type II models.
- Gauging risks and uncertainties.
- Minimizing risk occurrences.
- Finding policies as stable equilibria.

One strategy for integrating business thinking into network and system management is to view activities in terms of services, rather than as low-level protocol interactions. This has become a popular paradigm since the ITIL/BS15000[1] language of service provision and Service Level Agreements etc. See also the work in the Telemanagement Forum with eTOM and DEN-ng[2, 3].

Although the individual details of procedures are important, there is a more overarching need to understand the logistics of combining such procedures in the environment of an economic motor. We wish to turn some of these loose ideas into more rigorous ideas so that one can apply the techniques of scientific modelling and rational decision making to them.

Not all models that are related to aspects of a business are ‘business models’ in the fuller sense of the word. There are good reasons to separate off specialized issues and consider them separately. A business model claims to supply the raison d’être of a business, along with its capacity for long term profitability. Like a computer system, a business is a box surrounded by an environment with which it is interacting.

Modelling something as complex as a business in full is probably not a practical proposition, using today’s technology. Perhaps in the future one might model businesses as one today models the weather, but we are some way off having this technology today.

A key difference with computer management is that businesses tend to seek growth rather than stability.

We shall also looks at inventory models, efficiency models and so on, which pertain to smaller and more specialized parts of the whole.

The plan for this article is:

- Modelling business as a human-computer system.
- Identifying the principle for optimization.
- Managing internals, such as inventory and configuration.
- Planning high volume services and controlling uncertainties.

2 Heuristic goals of business

Whether explicitly defined or not, every business has a *business model* that defines the services, costs and strategies that allows it to survive in a competitive environment.

Service delivery is a key part of a business model. IT services are increasingly a part of this, but the principles for IT service provision are similar to those for the provision of any service.

Goal 1 (Service design) *A business must define the scope and quality of its services, ensure that they are deliverable and manageable, at the right cost.*

It is normal to speak of businesses as being service-based, in this discussion. This does not mean that supply or manufacturing industries are excluded, only that one should imagine supply businesses as providing a service to their customers. Thus the service paradigm is a convenient umbrella for business concepts.

Goal 2 (Capacity Management) *What rates of service delivery are required and how can they be achieved with a planned infrastructure.*

- A Service Level Objective (SLO) is the level of ambition that a provider aspires to.
- A Service Level Agreement (SLA) is a formalized agreement, i.e. a contract, between a provider and a client.

Goal 3 (Business relationships) *What is the structure of the organization with respect to suppliers and clients? Outsourcing must also be taken into account.*

Goal 4 (Risk and Security Management) *What are the risks of the business failure, either as a result of IT or other disasters, and what is the cost of securing against them?*

Goal 5 (Logistics of production) *What physical and resource constraints and are there to providing the service?*

Goal 6 (Arbitrary Business policy) *Decisions which cannot be made using a rational method of decision e.g. ethical or moral prerogatives.*

Quality control is a subject that applies identically to system administration and business administration. It requires a proper understanding of the operation of a system.

- A risk model for the system is required, so that one identifies the most likely threats to the success of the business or system (e.g. what happens if there is illness amongst key personnel, or a major power failure or natural distaster such as a flood?). Identifying these threats is important for the next point.
- Completeness of procedures: a response procedure should exist and be implementable for each event that can occur within the system. Responses to events should be mandatory and be executed within a minimum time-frame, determined by the scales identified above.
- Monitoring of performance: a non-intrusive method of monitoring performance should be available to all levels of the system or business, so that a natural feedback, with a minimum of overhead can be provided. System or Business Performance Indicators should address both internal procedures and external factors, which influence the markets for a business or the environment in which the activity of the system takes place.
- Securing security and efficiency in performance trough redundancy, load balancing, resource management etc.

3 De facto standards for human procedure

Industry standards exist for setting basic quality assurances for IT business[1, 4]. The standards are often presented as existing in order to protect business relationships such as outsourcing, in which a company isolates and gives autonomy to a part of its operation. Standards cover issues such as:

- Management's role
- Documentation requirements
- Competence acquisition (training and hiring)
- Configuration management
- Location and scope
- Monitoring and review
- Service Level Agreements with clients
- Complaints procedures: Fault recovery, Incident management, Problem resolution
- Budgeting and inventory
- Business relationships: dependencies (outsourcing)
- Legal constraints

The IT Infrastructure Library (ITIL) was developed by the Central Computer and Telecommunications Agency of the British government and has been adopted by many companies. It attempts to provide a ‘cohesive set of best practices’ for businesses and organizations. A British Standard which summarizes the issues in ITIL is BS15000[1] (see the contribution in this volume by Scheffel.). The IT Service Management Forum (ITSMF) is now the custodian of the standard[5].

The enhanced Telecom Operations Map (eTOM) is claimed to be the most widely used and accepted standard for business process in the telecom industry. The eTOM is analogous to and overlaps with ITIL and describes business processes required by a service provider as well as how they interact. eTOM provides more of a top-down hierarchical view of an enterprise whose aim is to provide service to external customers. There is considerable overlap between eTOM and ITIL, as is testified by the eTOM/ITIL interpreters guide[6, 7], but the eTOM vision is motivated very much by the traditional telecommunications hierarchy. It attempts to describe an organization as a database.

Management frameworks have been discussed by several corporate research milieux[8, 9]. The authors emphasize the difference between IT Management (service level management) and IT Governance (strategic planning and integration of IT in the business process). Information models for business interactions have also been discussed. [10, 3]

4 Agreements, contracts and relationships

Contracts are a tool for both establishing goals and imposing penalties. They are a method of codifying voluntary cooperation between peers in a social network[11]. Service Level Agreements are examples of contracts for service provision between IT peers.

Determination and optimization these contracts mixed both human and technological issues[12, 13, 14]. In particular, the context sensitive nature of agreements, and the great variety of automated technologies that involve contract exchange motivates ontological methods of harmonization[15].

Modal (deontic) logic has been used as a framework for making language of contracts precise, in a legal context[16]. Often, one is more concerned with standardized service level agreements in which one quotes well-understood specifications for average response time, minimum service level etc.

In ref. [10] contracts are used to define the penalty costs for failing to live up to their contractual obligations. This is reminiscent of the economical model of promise breaking[17]. A continuum study of this kind of contractual penalty feedback between a pair of peers has been examined in ref. [18]. There the authors noted the potential instabilities in relationships between agents is they were allowed to impose punishments and penalties on one another.

A further example of policy motivated by contracts is promise theory theory[19, 20, 21]. Here one considers the stability and consistency of a series of agreements in a network, to see whether the sum of individual promises can lead to a consistent union of ideas.

5 System, Environment and Scales

In the previous chapter in this volume[22], systems are described in terms of their degrees of freedom and their constraints. One should look for the freedoms, constraints and natural scales of measurement that are involved in the operation of a system.

5.1 Freedoms and constraints

There is a plethora of ways of modelling freedoms and constraints for a business model. Needs analyses can tell us about the kinds of variables at out disposal. The activities of customers and internal parts of the business are another way of identifying freedoms. Essentially the freedoms are the resources that are available to the enterprise, either directly or indirectly (e.g. by purchase or outsourcing).

Constraints on an enterprise include debts to be paid, sustainability or minimum profitability, physical and market limitations, equipment and processing etc.. "Use cases", in the sense of software engineering, can represent small subsystems within the whole. A quantitative model must show how freedoms and constraints should be balanced.

Most managers would like to live in a world that was clockwork and deterministic, however this is far from the case. The identification of random processes is important because this signals the extent to which the larger environment impinges on business activities. Random processes are essentially processes which cannot be fully understood without a complete model of the 'rest of the world'. Since such a model is impractical, one speaks of randomness, probabilities and expectations.

Event modelling is crucial in models and lead us to consider queueing systems, where service requests arrive as a stream of discrete 'orders' or units. Arrival and renewal processes are then needed to understand and optimize the strategies for dealing with demand[23, 24].

5.2 Scales

An identification of approximate numbers is the first step towards determining whether a business can sustain itself. The basic scales or numbers should be the average values, since most processes in a real world environment are stochastic in nature.

For example: what is the approximate demand for a service over a fiscal year? What is the rate of arrival of payments at different times of year? What are the estimated costs of running the business, e.g. the price per kilowatt hour of electricity, the rent of your data centre, wages for staff? What is the typical size and monetary value of a sale? How long does it take to complete and fulfill an order? How much profit can one make per order transaction? Given that short-term losses can be offset by long-term profits, over what timescale could one afford to make a loss? Give that there might be inhomogeneous demand and supply, over what time scale are the prices of services and resources approximately constant?

Comparing time scales is particularly useful way of seeing whether one has adequate resources to carry out a task. An organization that cannot deliver on its goals will not survive long.

If one thinks in terms of queues and arrival processes, we can pose the questions about service processing in a suitably analytical way: what is the time scale of trends and fluctuations in the arrival of service requests[25]? How large are fluctuations about the mean values and what is the scatter and jitter in them?

What total number of transactions can one expect in an interval of time e.g. a year? (A common model of sparse events is the Poisson arrival model, but this is known to work only for very large numbers of sparse arrivals, within the measurement interval, in general.)

Finally, from queue theory one ought to be able to estimate, what is the average response time of a business to a customer, or a server to a client?

In these questions, we already see that the same issues are directly applicable in both businesses and computer systems. This should not be a surprise, since the operation of a business becomes essentially machine-like once it has been defined.

5.3 Risk modelling

Fault tree analysis[26, 27, 28, 29] is a way of modelling the deviations from goals. It is a kind of anomaly detection and associated response. Anomalies are categorized into a tree of likely occurrences, and the overall risk can be evaluated as a probability. After this, the result must be translated into loss of revenues etc, by embedding the probable failure in the wider context of the income/cost model. What are the potential risks to loss of income or reputation? What is the contingency plan for recovery? The security management code of practice addresses this issue[4].

6 Business as a network service

The most important concept in a sustainable business process is that the value created by the process should be greater than or equal to the cost of operation.

6.1 A functional view of services

In a simplistic sense, a business model is something that explains the relationship between input and output. A business is a machine for generating wealth and produce (output), given a certain amount of investment and raw materials (input). The question is how it can be made self-sustaining (see fig. 1).

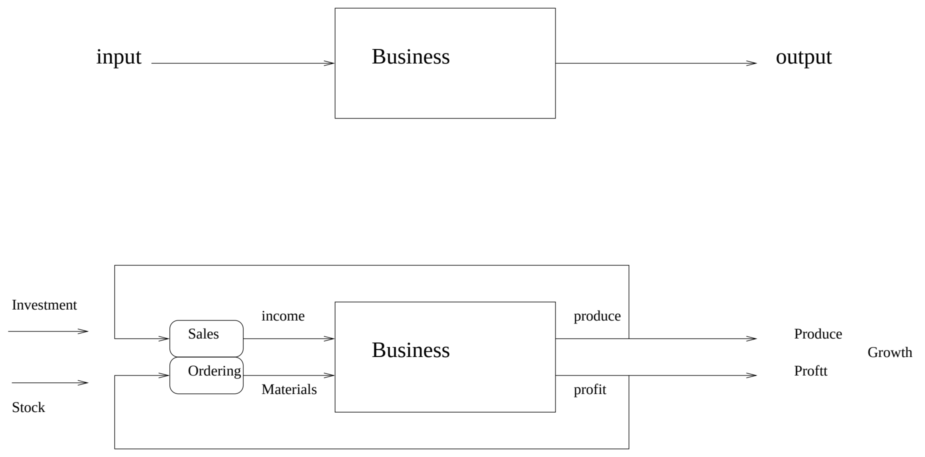

Figure 1: A functional viewpoint of a business as a self-sustaining process.

A self-sustaining process must have resources to begin with, and must produce more than it consumes. This is a peculiar idea, which seems to fly in the face of scientific reason and which readers may wish to ponder at length: how is it possible to sustain economic growth, and what is really being produced? Something from nothing? The answer, of course, comes down to understanding what value is, and why money is not a fixed scale of measurement, but rather a subjective convenience.

Let us set aside these issues and view the monetary value just as an arbitrary scale by which to express input and output. Then our simplest representation of a company is:

$$\text{output} = f(\text{input}) \tag{1}$$

which for the business translates in simplistic terms into:

$$\text{profit} = BusinessValue(\text{income}); \tag{2}$$

In terms of supply and demand, one inputs demand and outputs supply. This basic viewpoint of the business as a black box function with input and output is simplistic but essentially correct. There are various interpretations of this

Input	Output
income	payments
Materials	Production output

A more detailed modelling can be achieved by opening up and filling in the black-boxes, just as in software engineering.

$$(\text{profit}, \text{produce}) = BusinessResult(\text{income}, \text{materials}); \tag{3}$$

Then we take part of the profit to buy materials and sell the produce; thus functionally, we have a prototypical equation can be written in functional terms:

$$(\text{profit}, \text{produce}) \geq BusinessResult(Income(\text{produce}), Materials(\text{profit})); \tag{4}$$

We hope that this two dimensional mapping of produce and profit does not merely spiral to a fixed point of zero[30].

We do not normally control the input or output fully: it should be thought of as a random process[31]. Control factors such as advertising might influence expected demand at some predictable cost, but only within certain limits. Similarly production performance can be controlled by factors familiar in control theory[32], but only subject to basic limitations on certainty.

How shall we measure income? If the price of business services is constant then one could use either number of business transactions or the actual money paid into the business by external customers. But sometimes business give discounts for quantity etc, so there is a non-linear relationship between transactions and income.

Another issue is how should be coarse grain income? It has not traditionally been considered sensible to consider every individiual micro-payment as a separate event. Rather one collects income over an interval such as a day, week or even month and considers the gross amount for that interval as the income. Similarly with payments, this coarse graining allows a certain smoothing of the dynamical processes involved, allowing temporary debts to accumulate and be covered within a planning horizon. We can simplify this by writing with fewer assumptions.

$$BusinessResult(\text{transactions}) \geq Costs(\text{transactions}) \tag{5}$$

By including costs in the function on the RHS we can also allow for the fact that costs might not be linear in respect of production. Transactions is now a more or less pure variable that measures the demand.

6.2 Deterministic and stochastic demand

Demand is a stochastic process — an arrival process. Indeed, this simple black-box model is a queue. We might imagine that the cost of the business process includes certain economies of scale and therefore falls off to some kind of asymptote with increasing demand. Similarly the result from a small number of sales might be small and rise to a maximum asymptote with increasing demand.

Using linear programming, one can see this on a simple planar diagram (see fig. 2). Any business enterprise needs to incorporate this basic framework. Timescales are important in this interpretation however (see fig. 3). It might be feasible to operate at a loss for short intervals of time, provided we make up for this loss later on.

The arrival of business opportunities is an essentially inhomogeneous random process which one should consider smoothing into trends to eliminate such fluctuations. By smoothing, or local averaging one can look at the trends and use these for seasonal planning[33].

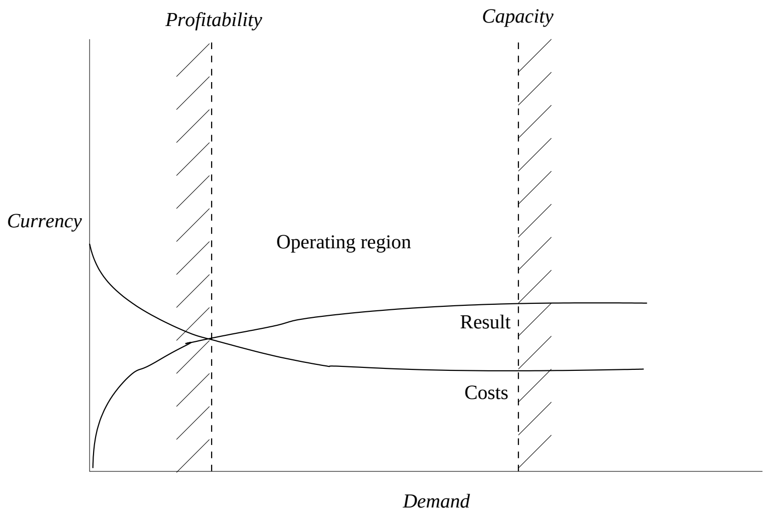

Figure 2: The operating region of a business lies in between profitability and capacity limits.

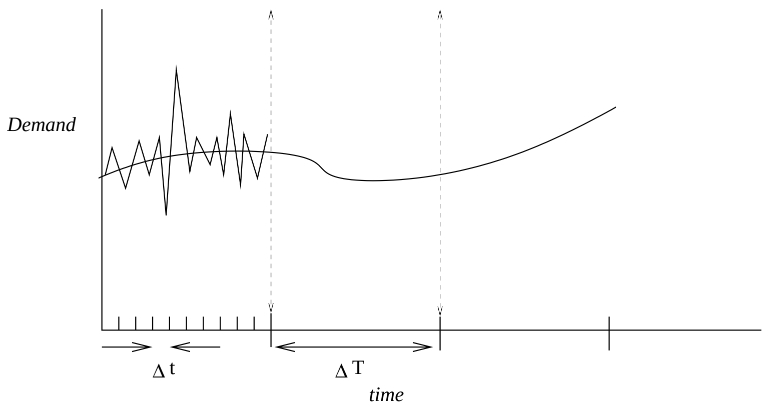

Figure 3: A time series view of the demand. The unpredictability of demand makes all its derived quantities random processes.

6.3 Diagrammatic modelling

The understanding of a system is greatly facilitated by diagrams and illustrations. Humans are particularly good at comprehending scenarios where several senses are involved. Visualization is one of the most important tools for comprehension.

Illustration generally begins by assembling the entities and relationships between them. There are many approaches to this, and one is unlikely to find a single diagram sufficient to describe what is happening in a business system. A more realistic expectation is to use a variety of diagrammatic techniques to model specific issues.

Two design strategies for building systems have emerged, and have developed with increasing refinements and compromises.

- *Top down*: The name 'top down' is motivated by the traditional way of drawing hierarchical structure, with high level (low detail) at the top, and increasing low-level detail at the bottom. A top down analysis is *goal driven*: one proceeds by describing the goals of the system, and by systematically breaking these up into components, by a process of *functional decomposition* or *normalization*.

- *Bottom up*: In a 'bottom up' design, one begins by building a 'library' of the components that are probably required in order to build the system. This approach is thus *driven by specialization*. One then tries to assemble the components into larger structures, like building blocks, and fit them to the solution of the problem. This approach is useful for building a solution from ‘off the shelf’, existing tools.

The difference between these strategies is often one of pragmatism. A top down design is usually only possible if one is starting with a blank slate. It might not be possible to implement one’s wishes from a top-down viewpoint with an existing set of constraints and components.

A ‘bottom up’ design is a design based on existing constraints, namely the components or resources that are to hand. An advantage of building from the bottom up is that one solves each problem only once. In a top-down strategy one could conceivably encounter the same problem in different branches of the structure, and attempt to solve these instances independently. This could lead to inconsistent behaviour. The process of ‘normalization’[34] of a system is about eliminating such inconsistencies.

- (Computer system - ‘top down’) In a the design of a new computer system, one examines the problem to be solved for its users (banking system,accounts), then one finds software packages which solve these problems, then one chooses a platform on which to run the software (Windows, Macintosh, Unix), and finally one buys the and deploys the hardware that will run those systems.

- (Enterprise - ‘top down’) In the organization of a maintenance crew, one looks at the problems which exist and breaks these down into a number of independent tasks (plumbing,electrical,ventilation). Each of these tasks is broken down into independent tasks (diagnosis,repair) and finally individuals are assigned to these tasks.

- (Computer system - ‘bottom up’) First one buys reliable hardware, often with operating system already installed, and installs it for all the users; then one looks for a software package that will run on that system and installs that. Finally, the users are taught to use the system.

- (Enterprise - ‘bottom up’) The enterprise bosses look at everyone they have working for them and catalogues their skills. These are the basic components of the organization. They are then grouped into teams which can cooperate to solve problems like diagnosis and repair. Finally these teams are assigned tasks from the list of plumbing, electrical work and ventilation.

In system administration, especially configuration management, these two strategies are both widely used in different ways. One may either implement primitive tools (bottom up) that are designed to automatically satisfy the constraints of a system, or one can use trial and error to find a top down approach that satisfies the constraints.

An entity-relation diagram, with associated data model, is a way of understanding information *types* and *relationships*, such as the approach used in the DEN-ng model[3]. However, such a diagram will not tell us anything about the efficiency of flows of work between the entities: for that, one needs something like a queueing network[24].

Transition diagrams provide a way of mapping out the activities in a system, in order to apply graphical methods of optimization[35].

This tells us how often the system spends its time the different states. By relating these states to the resources that are in use when the states are active, one can identify the most used resources during the interactive process. A centrality analysis of this graph can yield important information about resource allocation. For a deeper discussion of graph analyses, see the contribution by Canright and Engø-Monsen in this volume[36].

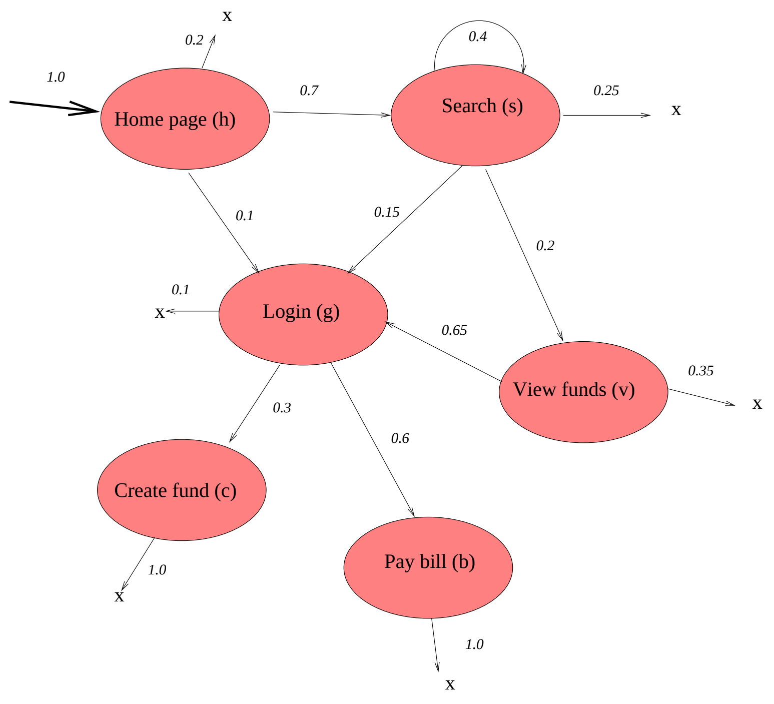

Figure 4: State transition diagram for a web based service. The transition probabilities weight the connections.

6.4 Continuum workflow models: scalability

Flow models take over where modelling of individual transactions becomes intractable. Continuous flows of information, goods or services are a convenient fiction for applying probabilistic methods. one replaces actual measurements with expectation values and events with probabilities.

Flow models use network or graph theory to analyse outcomes and consequences of flows. The six basic models of networking scalability are explained in [37]. See also the brief treatment in the foregoing chapter.

6.5 The hidden costs of ownership

A common way of expressing actual costs associated with components, in an enterprise, is the concept of *total cost of ownership*. Formally, this is the sum of all costs associated with the equipment, including the initial purchase price and licensing, over the lifetime of the equipment.

Let C be the number of customers, λ be the arrival rate of transactions, and P be the price of a transaction. Then we can say that, on average. the total income I, over a period is proportional to the product of these multiplied by the time interval Δt:

$$\text{Income } I \propto \langle C\rangle\langle\lambda\rangle\langle P\rangle\Delta t. \tag{6}$$

As usual, we use the notation $\langle X\rangle$ to mean the mean or expectation value of X. Thus business income from an online service will depend on the processing capacity[12]. Without proper modelling, it is easy to argue incorrectly that the correct strategy for equipment purchasing is to buy the fastest processor for the cheapest price. This is too simplistic, as we now show with an example.

Computer logic chips work by dumping large amounts of charge to ground through resistive materials, and because there is physical motion of mechanical components such as disks and fans, almost all of the power consumption of a computer ends up as heat eventually. Power consumption of modern computers is a major expense. Moreover, the cost of cooling the equipment is also proportional to the total power output of the environment, and thus to the transaction rate of data processing.

Recently several large companies have boasted the use of off-the-shelf PC or Macintosh hardware for high performance computing servers. This sounds like an idealistic and attractive idea, but the economics of using cheap equipment in data centres are not be as clear cut as one would believe. Measuring power consumption in a data-centre environment is a non-trivial task: power companies charge by delivered current, but that always over-estimates the actual power consumed, owing to *power factor* phase relationships in the alternating supply. Money can be saved even by investing in power factor correcting devices such as Uninterruptible Power Supplies, or more expensive hardware.

Inexperienced organizations think of the purchase cost rather than the cost of ownership, in production. Consider a simple example, based on numerical estimates from the Oslo region, here in Norway[1].

Let us suppose that we have a company considering populating a server farm with either cheap, off-the-shelf computer servers, or with high quality blade-racks.

Cost	Shelf Hardware	Quality Hardware
Purchase server	500 mu	1000 mu
Purchase switch	10000 mu	1000 mu
Power usage per server	0.1 kW	0.05 kW
Rent per 100 units per year	2400	600
Failures	15%	1%

One hundred servers, with associated switching equipment costs of the order of 60,000 mu for off-the shelf hardware and 101,000 mu for the quality hardware.

Electricity costs about 0.035 mu per kilowatt hour. Our cheap servers produce about 0.1 kilowatts of heat per unit, while the quality servers produce half this amount per unit. The cost of cooling is proportional to the heat produced. Thus it is proportional to the power consumption. Let us suppose then that cooling adds fifty percent to the total power consumption. Thus, in a data centre with a hundred servers, we are therefore paying electricity for a year, to the tune of

$$\begin{aligned} E_{\rm shelf} &= 100 \times 0.035 \times 1.5 \times 0.1 \times (24 \times 365) = 4599 \,\text{moneyunits} \\ E_{\rm quality} &= 100 \times 0.035.5 \times 0.05 \times (24 \times 365) = 2300 \,\text{moneyunits}. \end{aligned} \tag{7}$$

Thus the 41,000 money units saved on purchase cost leads to 2300 money units per year of additional power costs. It would take 18 years to justify the additional cost of these servers, by this reckoning. However, one must also add to this the cost of storage, cooling, management and reliability (replacements and repairs).

Cheap off-the-shelf PCs are large and bulky and need storage and cabling, and additional switching equipment. In additional rented infrastructure, one can expect at least:

$$\begin{aligned} R_{\rm shelf} &= 2400 \\ R_{\rm quality} &= 600 \end{aligned} \tag{8}$$

Failure costs, equivalent to 15 of the cheap PCs per year occur, but only 1 of the more expensive machines fails, so

$$\begin{aligned} F_{\rm shelf} &= 15 * 500 = 7500 \\ F_{\rm quality} &= 1 * 1000 \end{aligned} \tag{9}$$

[1]The costs are approximate and based on Norwegian kroner divided by a factor of ten for more ready comparison with dollars and Euros.

The total 'saving' by buying off-the-shelf hardware ($\Delta =$ quality minus shelf) is thus

$$\begin{aligned}\Delta S &= \Delta R + \Delta F + \Delta E \\ &= -1800 - 6500 - 2300 \\ &= -10600 \mathrm{mu/year} \qquad (10)\end{aligned}$$

In $41000/10600 = 3.9$ years, one has recovered the cost of the quality servers. This could be the actual lifetime of the cheap off-the-shelf PCs, thus, it is not clear that one actually saves any money by buying cheap commodity hardware.

We have not included the costs of administrating and lost revenue due to down-time etc. There are additional costs involved here which could tend to favour more expensive hardware. Soon, the rate of development in chip speeds etc will slow down and environmental levies will be placed on the disposal or recycling of computing equipment and this cost will also be paid by the consumer. Thus the present style of throwing away equipment after a few years will eventually have to change. Investment in quality could be the strategy of the future.

7 Demand and capacity: 'provisioning'

The Americanization 'provisioning' has become common parlance for the activity of providing the necessary utilities to cope with demand, i.e. the provision of service capacity.

All we can say with certainty, about a system, is that the rate of production in the business cannot exceed some maximum capacity C. If a business is merely a fixed machine whose costs are fixed, then the capacity of the system can be fixed. The problem then is to maximize capacity to cope the maximum expected demand. (see fig 5) This is often done by arranging for a certain over-capacity. However, if costs are proportional to some monotonic function of capacity then one could envisage an enterprise saving money by adjusting their available capacity (re-deploying equipment, hiring and firing etc). See fig. 6.

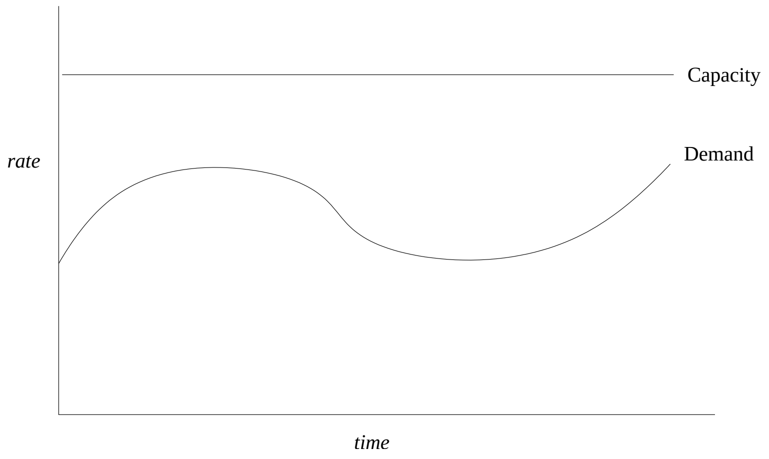

Figure 5: An over capacity—does this gap between capacity and demand cost anything for the company?.

8 Service provision models

Network services are the new paradigm for almost every new enterprise and service today. Even production industries can be regarded as services producing goods within a network

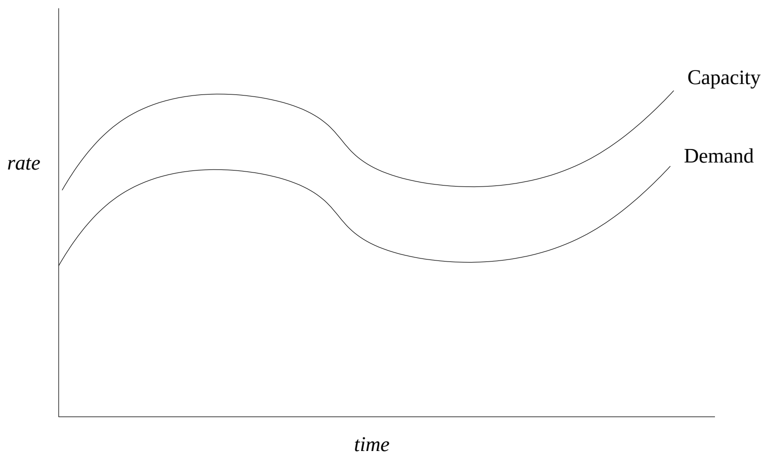

Figure 6: Capacity adjusted to follow demand—in principe more efficient.

of suppliers and consumers. The service model is simply a way of thinking. The advance which accompanied this point of view was the idea that delivery of a service is a contractual agreement between parties, and that customers are allowed to expect a certain Quality of Service (QoS).

Quality levels are negotiated using Service Level Agreements (SLA). In the literature, these concepts have multiplied into a plethora of Three Letter Abbreviations, including Service Level Objectives (SLO), Service Level Indicators (SLI) etc. A service level agreement is a contract, and thus one imagines that failure to live up to the expectations of the agreement will lead to reprisals, such as unpaid bills or lawsuits.

One is therefore interesting in dimensioning service capacity to be able to fulfill the contracts that have been signed. There are various approaches to this.

- Over-provisioning: by making service capacity much better than demand, one allows for any margins of error by always having extra capacity. This implies a certain level of waste.
- Optimization: calculating the optimal resource needs using a models and data measurements.
- Inventory management: use of inventory control methods to provide a more dynamic resource management based on periodic maintenance of stocks (this is somewhat analogous to periodic system maintenance regimes).

The service paradigm allows us to couple models to well-known analyses like queueing theory, which deals with everything from packet networks to vehicle traffic.

In terms of business, or enterprise organization we would like to have some way of calculating reasonable boundaries and levels for service, based on what we know of our capabilities. An interesting approach to this has been suggested by Sauvé et al. [12]. There one considers a Service Level Agreement to consist of two essential items per service type:

- A minimum service level (availability)
- An average response time $\langle t \rangle$

In addition two input values are required: a longest response before users ‘defect’, or give up on delivery and go elsewhere, and a probability distribution for the response time in relation to the threshold, so that the probability of being over threshold can be computed. Using this basic information, one can create a cost model for service provision, including over-capacity. By minimizing a cost function, one can determine the optimum number of servers required to support a given Service Level Agreement, or the maximum values in an agreement that can currently be supported by current hardware.

8.1 Quality of Service

The concept of Quality of Service (QoS) has moved centre stage in network management as service providers seek to offer measurable guarantees to their customers in the form of Service Level Agreements (SLA). The Quality of Service builds on terms like QoD (Quality of Devices), QoE (Quality of Experience), QoB (Quality of Business), and any number of variations to discuss the issue of service provision. Each of these is trying to capture the essence of a usable measure, or 'value', that can be sold to customers. It has been suggested that service quality must be a function of 'Quality of Devices', since service is a function of devices:

$$\mathrm{QoS} = f(\mathrm{QoD}) \tag{11}$$

This makes clear sense: it follows from the laws of causality. The question remains, however, what kind of function should this be, and would a knowledge of the function help us to understand the limits of predictability of service levels?

Quality of Service advertises a qualitative characterization, where one would prefer a quantitative one. One must therefore pin down the meaning of the term.

The key measurable to be average service rate R, measured in *bits per second*. We interpret Quality of Service to mean the level of predictability in the *quantity of service*; i.e. we identify:

$$\text{Quantity of Service} \pm 1/\text{Quality of Service} \rightarrow R \pm \Delta R. \tag{12}$$

Thus, if quality of service is high, the uncertainty in service level is small and guarantees will be reliable; and if it is low, the uncertainty is high and guarantees are likely to be unreliable. This viewpoint ties Quality of Services to more traditional quantities, such as reliability and certainty.

Ref. [13] also attempts a causal approach. This paper captures the spirit of a causal approach by suggesting the link between service quality to its dependent variables, but it does not take the idea to its logical conclusion by actually defining the functional relationships or how they relate to the service quality. In ref. [38], a heuristic overview with similar ideas was presented, and in ref. [13] an acceptance of stochastic ideas was acknowledged in modelling using network simulations.

The method of Fault Tree Analysis[29, 26] can be used to construct an overview of the causal processes. The theory of errors and uncertainties[39, 40] may be used to estimate the probabilities.

Fault tree analysis can be used to gauge both the likelihood of error and the uncertainty in service provision (see fig. 7). Probabilities can be used to represent either the likelihood of a service failure (i.e. the likelihood that an SLA is not fulfilled), or the probability of deviation from expected (average) service levels. In the latter case, there this gives us a way of deriving the limits of certainty in service provision.

For instance, suppose we have – at the very lowest level of the system – that the data rate for the transmission medium is given by Shannon's Gaussian noise formula:

$$C(B, S, N) = B \log_2 \left(1 + \frac{S}{N}\right), \tag{13}$$

where C is the channel capacity in bits per second, B is the bandwidth in Hertz, S is the signal power and N is the noise power. We now wish to discuss the effect of temperature

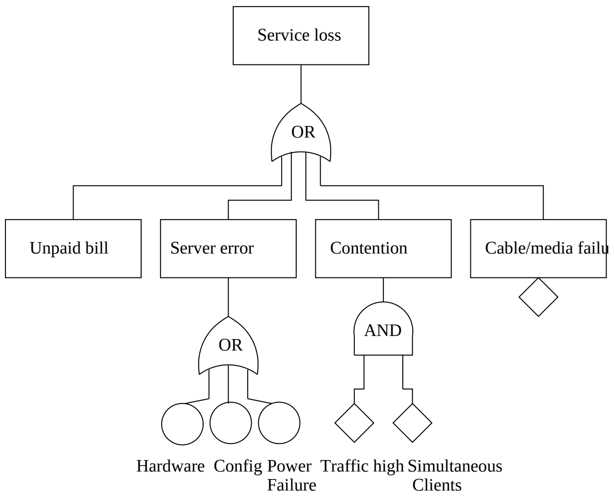

Figure 7: Fault trees can capture the essence of the hierarchical relationship between quality of components and final service. Here is a typical example of how fault trees are used to gauge reliability. Logical AND and OR gates reflect the interdependency of lower level components in a system.

on the capacity of a metallic conductor (copper wire), for which it is known that thermal noise has the form

$$N_{\rm metal}(T) = N_0(T + 273), \tag{14}$$

where N_0 is a constant and T is the temperature in Celsius[2]. The complete formula is given by straightforward substitution:

$$C(B, S, T) = B \log_2 \left(1 + \frac{S}{N_0(T + 273)}\right). \tag{15}$$

With the full expression capturing all of the dependencies in the dependency tree, through substitutions, the theory of errors and uncertainties can now be applied to the list of independent parameters.

8.2 Non-linearity: reactionary models of service dependency

Service provision is, in some contexts, thought of as the delivery of fixed-rate, policy-controlled transactions between willing parties. A contract of service, called a Service Level Agreement (SLA) documents the promise that the service provider makes to the client. The SLA is a form of policy for the server to uphold[41, 42]; what distinguishes it from other policy rules is that it is not deterministically enforcable, since the parties have no direct control over one another.

Any service provider's policy is subject to environmental uncertainties and, for that reason, the promise can never be more than an expectation or hopeful prediction of average service levels[43]. To verify that a promise has been kept, the client must monitor the service level. Service Level Agreements are often complex. They combine several measured values into composite metrics that are then compared to a template[13, 14, 44, 45, 46]. This has important implications for the stability of policy relationships.

[2]Note that the uncertainty due to thermal noise is not negligible in high capacity copper wires: it can represent tens of Megabytes that might be sold to a customer.

Feedback regulation of policy is an idea that has received more attention recently, due to interest in autonomous system operation; it has been discussed in a variety of circumstances[47, 48, 49, 50] for different service types ranging from web services to configuration management. Feedback regulation occurs regardless of whether it is carried out autonomously by machines or by humans, whenever there is a dynamical interplay between monitoring and SLA.

From the theory of dynamical systems, it is known that the combination of several dynamical variables using AND (multiplication) and OR (addition) in feedback loops can lead to 'chaotic' or non-linear behaviour. Such combinatoric ideas have been discussed to provide more expressive policies for service level agreements[13, 14, 44, 45, 46]; for example, in ref. [13], the authors discuss 'aggregated Quality of Service parameters' of the form:

$$S \text{ depends\_on } (SS_1 \textbf{ AND } SS_2) \tag{16}$$

as well as more general rules like:

$$QoS_A(S) = f(QoS_B(SS_1) \textbf{ AND } QoS_B(SS_2)). \tag{17}$$

They are basing service quality promises on a function of measured values. However, there are problems with combining dynamical variables non-linearly on finite precision systems. It is known that such systems can spiral out of control, or simply ebb to zero for certain ranges of parameters (for a review, see [51]). It is therefore important to understand how such behaviour emerges in dynamically regulated policies and what can be done to secure stable operation.

In ref. [18], the authors attempt to answer the basic question: whether or not a reactive policy, based on mutual observation of several data points, can lead to stable behaviour or whether escalation of reprisals by the parties can spiral out of control. The studies made on continuum approximation models indicate that feedback regulation of SLA-like policies is a highly complex matter that must be parameterized with caution. Even simple non-linear policies have no stable fixed point that can be associated reliable service levels, other than the trivial case of no service. If two parties enter into a policy agreement of this type, they will either undergo a long fibrillation of changing behaviour, or they will spiral down to offering zero service levels. Thus the automation of policy should be wary of these issues.

9 Inventory models

Most traditional businesses involve the actual sale of goods which need to be stored in some kind of warehouse. The name *inventory* is used for stocks of revenue generating goods. For example, an Internet bookseller, Like Amazon, needs a warehouse to keep its stock, pending retail sale to customers. There is a cost involved in storing goods, both in terms of expected returns and in terms of warehouse rent. The longer inventory is sitting in a warehouse, the less profitable it can be.

However, even a computer centre needs certain supplies to function. Computers, monitors, mice, keyboards, CDROMs, hard-disks, network cables etc, are all examples of potential inventory that a system administrator should consider keeping as inventory.

There are three considered motives for storing inventory, rather than ordering on demand (or so-called Just In Time) models[52].

1. Transaction motive: by ordering large or predictable quantities of the merchandise, one can minimize transaction costs, or administrative overhead k.

2. Precautionary motive: Keeping inventory is a way of minimizing the risks of uncertainty. It provides a buffer against late deliveries or sudden demand.

3. Speculative motive: If the price of stock is expected to rise in the near future, pre-ordering and stockpiling can be a way to save money.

Modern computer systems do something similar in their ‘caches’, where the cost in not immediately monetary but a cost in time. Recently some authors have attempted to map inventory models onto service based computing models[9]. Although these issues are motivations for inventory models, the models themselves only account for the first of these reasons.

Inventory models are essentially primitive form of queueing theory, where the focus is slightly different. They were invented to deal with actual warehouses (see the reviews in volume of ref. [52] for lucid introductions) and the logistics of planning, but they can be applied to Internet commerce and also elementary transaction performance models for networking. The over-simplifications involved in the most basic models are usually tolerated precisely because of their simplicity.

9.1 Basic continuum model of averages

The basic model of inventory is known as the Economic Order Quantity (EOQ) model and assumes a completely deterministic process of predictable supply and demand. While this is clearly not true, by any stretch of the imagination, it is plausibly true over long times for average quantities. It can therefore be thought of as a ‘mean field approximation’.

We make the following assumptions:

1. The inventory of merchandise is all of a single type and stored at a single location.
2. We expect inventory to cycle continuously with a provision of service that never stops (a deterministic queue). So we ignore start up costs, and we are interested in the costs per inventory cycle-period, i.e. the time period T of the inventory cycle is a natural timescale for the model.
3. The model is deterministic, i.e. the rate of consumption (or the *expected demand*, in a sales interpretation) for merchandise is known with certainty, and is λ inventory units per unit time. It is like a leaking bucket.
4. One cycle of the model is one single (average) order of (average) size q, taking (average) time t.
5. Shortages of inventory do not occur, nor does over-production, i.e. demand is always matched to supply, on average.
6. Delivery time is instantaneous, i.e. immeasurably short compared to an average timescale T for ordering.
7. We do not take into account the provision for discounts for old stock, i.e. merchandise is as valuable today as it was when it entered the inventory.

As always, the assumptions underpinning a model are the key to knowing how and when we should take its results seriously.

Let us assume that each order of goods is made with average quantity size q, which is clearly constant over the time interval for which the average is defined. The cost of each order, of size q, has a fixed average transaction price k (money units) for and an overhead that comes from the cost rate of *holding* the merchandise in the warehouse h (money units per unit time) over time.

Finally, let the actual inventory, as a function of time be written $I(t)$ (see fig. 8). From our constraining assumptions, the inventory function can only have the simple form:

$$I(t) = q - \lambda t, \quad (0 \leq t \leq T). \tag{18}$$

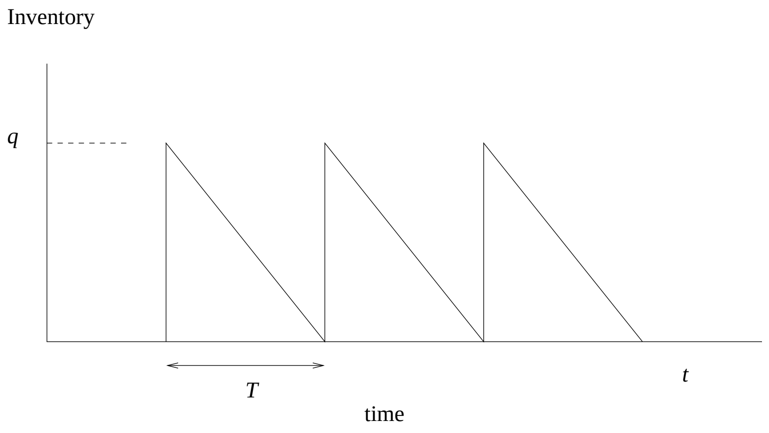

Figure 8: The basic inventory model with continuous cyclic operation shows how the stock is produced 'suddenly' and used at a constant rate λ until it is exhausted.

The cost per unit time of keeping merchandise is

$$C(T) = \frac{1}{T}\left(k + h\int_0^T I(t)dt\right) \tag{19}$$

and it measured in dimensions of money units per period. From fig. 8, we note that $T = q/\lambda$, just from the geometry of the triangles. This integral can be calculated trivially, or one can use the half-base-times-height rule for the area of a triangle:

$$\begin{aligned}\int_0^T I(t)dt &= (qT - \frac{1}{2}\lambda T^2)/T \\ &= \frac{1}{2}q^2\lambda \\ &= \frac{1}{2}qT. \end{aligned} \tag{20}$$

Thus, we have a cost per cycle/order function that may be expressed either as a function of q or of T:

$$C(T) = (k/T + \frac{1}{2}h\lambda T) \tag{21}$$

$$C(q) = (k\lambda/q + \frac{1}{2}hq). \tag{22}$$

In either variable, we see that the cost per order is a non-linear function that is the superposition of a hyperbola and a straight line (see fig. 9).

The cost function has a minimum, which may be found by differentiation by q or T, at the values q^* and T^* respectively:

$$q^* = \lambda T^* = \sqrt{2k\lambda/h}. \tag{23}$$

at which point

$$C(q^*) = C(T^*) = \sqrt{2k\lambda h}. \tag{24}$$

Thus, if we have optimized the average performance of this systems, the average cost per order grows with fixed order price with the holding cost.

The result displays a symmetry between order cost and holding price and demand:

$$\langle k, \lambda, h\rangle \tag{25}$$

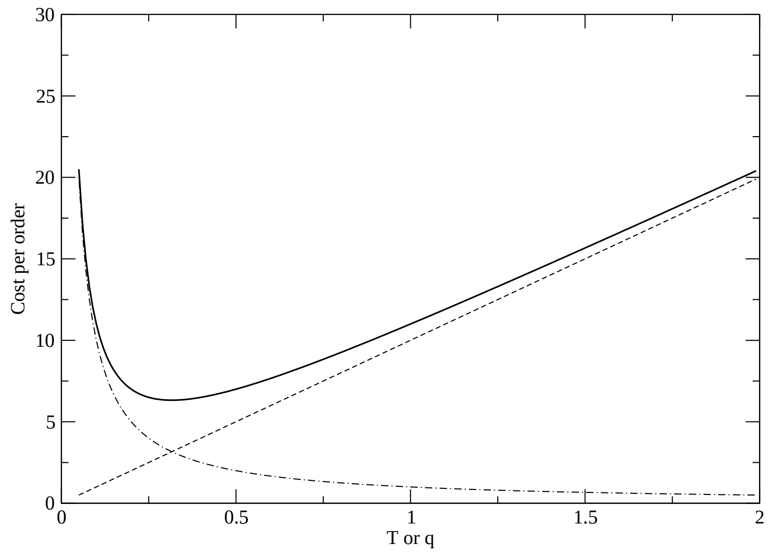

Figure 9: The solid line cost per order (or per cycle) function, is the superposition of a linear growth and hyperbolic cost per unit time. In either parameterization, has the form of a lifted hyperbola. There is a clear minimum at an abscissa point T^* or q^*, which represents the minimum cost value.

How do we explain this symmetry causally, and what does mean?
We must first explain the interpretation of the variables, and understand their forms.

- T^* is the cheapest time interval for customers to place their orders. If we make T less than this, eqn. (19) tells us that the transaction price will dominate the cost, i.e. the customers will pay the basic processing fees too often. If we make the time larger than this, the holding time dominates the cost, as the storage cost of the inventory increases and is passed on the the customer.

- q^* is the cheapest order size. It is related to the time by the constant demand rate, so they are not independent variables. If q exceeds the optimum, i.e. customers greedily order too much at a time then the order period increases, and vice versa.

The optimum cost increases as either the unit cost, storage cost or demands increase. This is because the larger costs force the consumer to slow down the rate ordering (increase T^*).

The steady-state, average assumptions of the model allow us to eliminate time as a variable. At optimal operation.

How do we know that we can tune the system to perform optimally? On average we can imagine that it is reasonable to compensate for non-optimal during a period by suitable monitoring.

Suppose we do not manage to compensate, a deviation is characterized by the beautiful dimensionless relation:

$$\frac{C(q)}{C(q^*)} = \frac{1}{2}\left(\frac{q}{q^*} + \frac{q^*}{q}\right). \tag{26}$$

Suppose we achieve $q = \frac{1}{2}q^*$, then this ratio is 1.25. In other words, missing the target by a hundred percent of th achieved goal leads to only a twenty-five percent increase in costs. This is quite insensitive.

9.2 Applying the model

The inventory model is not a complete business model, in the sense of dealing with income and expenditure. It is a sub-model that deals with optimizing the costs of holding stocks and supplies. The burden of applying these formulae, as always, is in finding reliable and realistic data for the inputs.

The optimal rate of ordering is not about how much the inventory costs to purchase. It is about the administration costs. This, if P is the purchase price per unit for inventory, the total cost of the order is simply qP, but the cost of ordering and keeping the inventory $C(q)$ is an overhead which is to be minimized.

- k is the administrative cost of placing an order, including the actual time taken converted using the man-hour rate.
- h is the cost of keeping inventory which includes storage, insurance, any interest paid over the period on money borrowed etc.

1. Suppose that the yearly consumption of PCs is about 20 per year. An order of PCs, which includes decision-making, discussion, and processing accounts for 5 man-hours of time at 50 money units per hour, hence $k = 250$ money units.

   The cost of storage for these PCs, including cleaning and insurance per unit, per year is $h = 100$ money units per unit per year.

   The optimal order size is now,

   $$q^* = \sqrt{\frac{2 \times 250 \times 20}{100}} = 10 \text{ units} \tag{27}$$

   and

   $$T^* = \frac{1}{2} \text{ years.} \tag{28}$$

   Thus, two purchases of ten units, over a year is the optimal solution.

2. For a second example, consider the consumption of disk space on a mail server. As users accumulate E-mail, or as an ISP accumulates customers with fixed quotas. (In fact dealing with fixed quotas is much easier to analyze because we can more easily estimate the consumption, without having to take into account how much users tidy up old mail etc).

   We shall take disks to be the unit of inventory. Suppose the expected number of new customers (consumers) per year is 100 per year. If we assume that a typical hard-disk can accommodate 100 users, then this amounts to a usage of $\lambda = 1$ disk unit per year.

   The processing cost for an order of disks is a half man-hour at 50 money units per hour, i.e. $k = 25$.

   The holding cost for disk space (assuming that they are installed on arrival) includes power consumption, replacement disks on error etc. Suppose this amounts to a small amount of power consumption, plus a ten percent chance of having to buy a new disk at 500 money units. Let's take $h = 55$ money units per disk per year as the holding price. In fact, if we allow for the rapidly increasing capacity of disks, we might make h larger to account for the loss of disk space from ordering too early.

   The optimal order size is now:

   $$q^* = \sqrt{\frac{2 \times 25 \times 1}{55}} = 0.95 \text{ units} \tag{29}$$

   This tells us that there is no benefit to mass ordering disks before they are needed. It is okay to order them one at a time, since they are relatively expensive to own, and relatively cheap to order.

9.3 Extended model

In reality, the production of merchandise takes a certain amount of time. to produce or assemble the inventory. We can make a trivial modification of the model to include a finite and linear production schedule, with the simplifying assumption that the rate of production ψ is always great than the rate of consumption λ.

We shall once again assume an order size of q and a cycle time of T. By analysing the

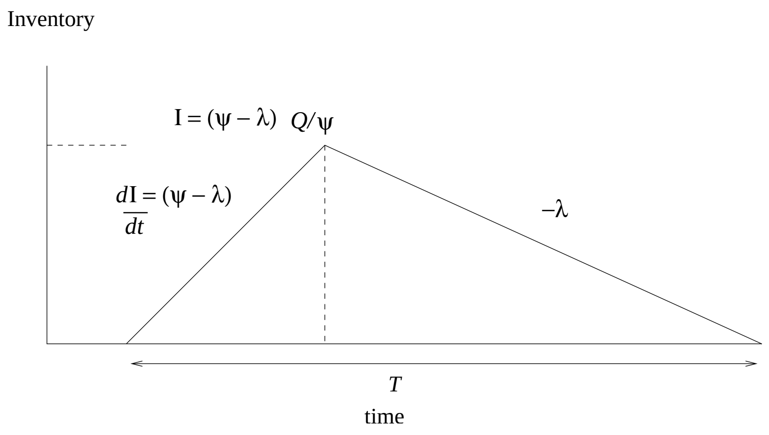

Figure 10: The extended inventory model with continuous cyclic operation shows how the stock is produced at a finite rate ψ and used at a constant rate λ until it is exhausted.

triangular period in fig. 10, we can use the triangle formula to find the area, as before in the integral of eqn (19). The vertical height of the triangle (representing the peak amount of inventory in store, given the stock is being used as it is being produced) is

$$I_{\max} = (\psi - \lambda)q/\psi. \tag{30}$$

The length of the base is still T. Moreover, by examining the triangle, we may confirm that $T = q/\lambda$ still holds.

The cost per order function is thus now:

$$\begin{aligned} C(q) &= (k + \frac{1}{2}h(\psi - \lambda)Tq/\psi)/T \\ &= \frac{k\lambda}{q} + \frac{1}{2}h(\psi - \lambda)\frac{q}{\psi} \end{aligned} \tag{31}$$

Minimizing this cost function, by differentiation gives an optimal order size:

$$q^* = \sqrt{\frac{2k\lambda\psi}{h(\psi - \lambda)}}, \tag{32}$$

and we recall that $\psi > \lambda$. As $\psi \to \infty$, this model become the basic model.

9.4 Effect of uncertainty in the rates

Even in an averaged regime, one must expect a certain amount of uncertainty in the rates of production and consumption. Suppose we assume that ψ and λ are independent variables subject to small, symmetrically distributed uncertainties:

$$\begin{aligned} \lambda &\to \lambda \pm \Delta\lambda \\ \psi &\to \psi \pm \Delta\psi \end{aligned} \tag{33}$$

Then, we may calculate the compound uncertainty in q^* by the Pythagorean combination of the first order Taylor expansions, in the usual way.

$$\begin{aligned}\Delta q^* &= \sqrt{\left(\frac{\partial q^*}{\partial\lambda}\right)^2 (\Delta\lambda)^2 + \left(\frac{\partial q^*}{\partial\psi}\right)^2 (\Delta\psi)^2} \\ &= \frac{1}{2}\frac{q^*}{|\psi-\lambda|}\sqrt{\left(\frac{\Delta\lambda}{\lambda}\right)^2 \psi^2 + \left(\frac{\Delta\psi}{\psi}\right)^2 \lambda^2} \qquad (34)\end{aligned}$$

There is a similarity between the idea of inventory control and the idea of system maintenance. The maintenance theorem of ref. [43] says that a system is maintainable if

$$\frac{1}{\langle q\rangle}\frac{d\langle q\rangle}{dt} \ll \frac{1}{T} \qquad (35)$$

where $\langle q\rangle$ is the average level of inventory.

However, we note that the inventory models are steady state models, so the maintenance theorem is satisfied automatically, since $\langle q\rangle$ is constant, by assumption. If, however, we allow for stochastic uncertainties in the model, then eqn. (35) becomes essentially

$$\frac{1}{q}\frac{\Delta q}{T} \ll \frac{1}{T} \qquad (36)$$

which translates into $\delta\lambda/\lambda \ll 1$ and $\delta\psi/\psi \ll 1$.

10 What is the cost of poor system administration?

There are clearly many areas in which a system administrator's decisions affect the potential for a business to operate. In purchasing decisions and operating costs, there are large amounts of money to be saved. In a globalized environment, in which one has data centres with load balancing, companies could even consider re-routing requests to locations where load is low or power is cheap. Resource utilization management is an important part of cost management.

A competent engineer is one of the most important accessories to any machine or system. It is usually the member of personnel who is most forgotten until needed. Delays in an enterprise's operations could mean lost revenue, lost business or even lost reputation (an ability to attract new income). The repercussions can be serious.

In the present world of virtual issues like CPU speeds and numbers of megabytes, one sometimes forgets the importance of physical issues. What is the effect of heat? How should one avoid earth loops in a server room? Don't hang the leaky air-conditioner over your expensive mainframe, and so on.

10.1 Importance of empiricism

In the second world war, pilots were said to fly by the seat of their pants, meaning that they were intimately connected to their machinery—that they had learned the bumps and foibles of the machine. An understanding of the empirical nature of a system is just as important today, even if there are no physical vibrations.

How does a system respond under load? How much power does it use? How does the weather affect the performance of computing equipment, or the staff? While these might seem like frivolous questions, they have a very serious side.

In a data centre, the questions above become design issues. A system might thrash and slow down under heavy load. A peak of power consumption might cause fuses to blow or cooling equipment to overload. The weather can affect the air in the centre: if the humidity is too high, condensation can form on cool surfaces; if it is too low (very cold weather), static electricity can cause errors and even power spikes that can damage equipment.

Perhaps we might think that running a server room a few degrees hotter can save some money in cooling power? Or that extra humidity can increase the specific heat capacity of the air and increase the efficiency of the eat transfer? These things can only be understood through rational inquiry and measurement. For example, the specific heat capacity of dry air is 1005 Joules per kilogramme per degree Kelvin, and the heat capacity of air with one hundred percent humidity is 1030 J/kg/K. The difference is only one percent, which is probably far less than normal power fluctuations, and is therefore irrelevant. On the other hand, humidity should be kept between forty to fifty percent to balance the risks of condensation and static electricity.

The scientific method: rational inquiry is a powerful medicine against such problems.

10.2 Return on investment and total cost of ownership

Return on investment is defined as the income per fiscal year, divided by the cost of the investment. This is often proportional to the throughput of the enterprise. Ineffective use of resources (poor scheduling or load balancing) can limit this. Poor training of staff and inadequate management can also affect it. Automation of basic tasks can increase the up-time of the system and avoid security problems.

10.3 Risk minimization

Understanding the risks of an organization and being able to plan for them is only possible with analytical tools. The identification of points of failure in the system network, the analysis of dependencies and probability that service targets will be met: these are examples of issues that are impossible to plan for without a mixture of empiricism and analysis.

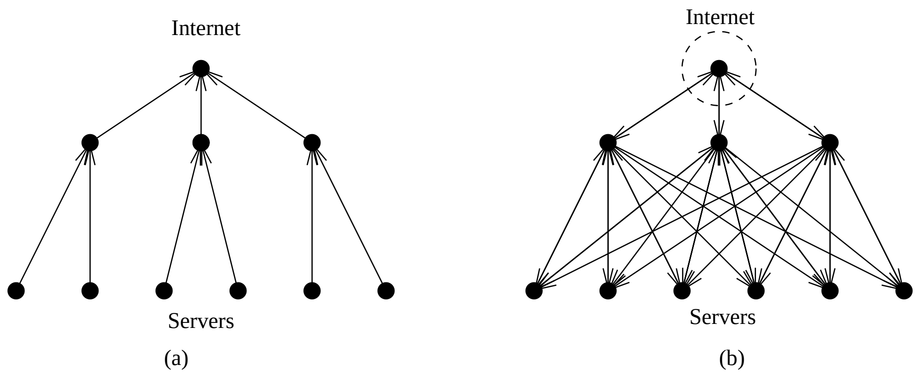

Figure 11: Load balancing without points of failure is a tricky matter if one is looking for redundancy. A load balancer is a tree – which is a structure with many intrinsic points of failure and bottlenecks. How can these be completely eliminated to make a high availabilty data centre?

Risk can be used a means of optimization. For example, in ref. [53], the authors use a minimum risk principle to find the best time to make a backup copy of data in a busy data centre. Many administrators will do this when the system is most quiescent; however, an analysis of risk shows that that is not the way to minimize risk of loss due to random error. Rather, the answer is about halfway between peak activity and minimum activity (after peak time), since more changes that can be lost occur during peak times (see fig. 12). Whether an optimal solution is desirable is a matter for policy.

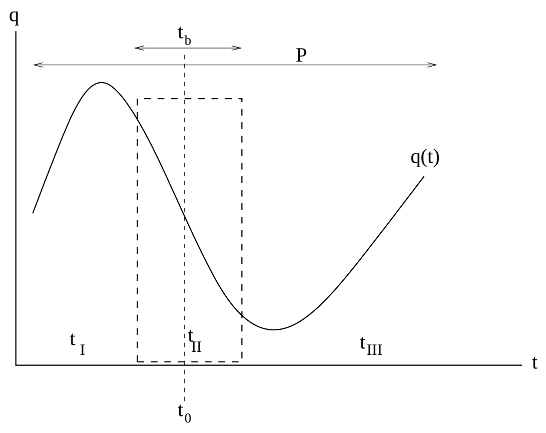

Figure 12: The 'windshield wiper model' for disk backup scans across files in a time t_b about t_0. The wavy line shows the average rate of point change arrivals during a 24 hour period. There are three regions of currently unknown sizes: region I before backup, region II during backup and region III after backup. The file change process $q(t)$ brings q arrivals at time t and provides a probabilistic weight to the expected risk: if more files are arriving, the risk of loss will be higher.

10.4 Incident response and fire-fighting

Incident response is like an inventory model or queueing process. Incidents or events form an arrival, point process that accumulate into a queue or 'inventory' of tasks to be cleared. By analogy with the language used in configuration management policy, a business 'incident' could be defined as a deviation from policy, or it could be a complaint or a failure of some part of the business to deliver on a promise. We can model this kind of process with various levels of sophistication.

The illustration in fig. 8 can also be interpreted, somewhat simplistically, as a deterministic expectation of maintenance costs, At the leading edge of the saw-tooth one can imagine a quota of jobs to be complete arises (suddenly). Gradually, these jobs are completed and the inventory is cleared. After a periodic interval, one reschedules the next batch.

Although simplistic, this could be a realistic approach to time management—a way of clearing processes from a help-desk message queue, or some other E-mail queue. One simply chooses $\lambda = 1/T$ to fix the frequency of turn-around. There is an initial logistical cost of scheduling time to answer job messages (e.g. learning and research time to study the problem), and there is a cost associated with having them unanswered, or un-repaired (broken promises, maintenance contract penalties).

The optimum time T^* tells us how to prioritize this process; the optimal inventory order tells us how many jobs q^* to schedule at a time. This is related to the queue length in queueing models.

Another example of this kind of scheduling is periodically scheduled maintenance tasks, like garbage collection and configuration maintenance. where the scheduling order cost is the amount of resources needed to locate and fix potential problems. The holding cost is harder to estimate, but involves lost productivity or due to faults.

Incident management in relation to ITIL and business objectives is advocated in ref.[8]. The authors consider a policy-based prioritization, based on the penalty costs of failing to meet obligations in Service Level Agreements, and using time-discounting as an importance ranking, based on the urgency of the incident. It is a form of anomaly detection. These authors take essentially the analogous view to anomaly definition as is put forward for system administration in refs. [43, 54], namely that policy is an expression of the limits of a probability distribution. Prioritization can be defined essentially by the number of standard deviations off 'normal'

Policy involved defining a series of mappings from low level events into a policy model. Thus policy involves an associative map, based on a notion of average correctness and a classification of incident 'distance from normal'.

11 Conclusions

How does system administration integrate into the business process? It is centre stage. The system administrator plays the role of a service provider and of a maintenance agent. In a world of increasing reliance on computing systems, the engineers who design, tune and maintain the system are the keys to its success.

References

[1] British Standards Institute. *BS15000 IT Service Management*, 2002.

[2] Telemanagement Forum.

[3] J Strassner. *Police Based Management, Solutions for the Next Generation*. Morgan Kaufmann/Elsevier, 2004.

[4] British Standard/International Standard Organization. *BS/ISO17799 Information technology – Code of practice for information security managementt*, 2000.

[5] Information Technology Service Management Forum. *http://www.itsmf.com*.

[6] Telemanagement Forum. An interpreter's guide for etom and itil practitioners, 2004.

[7] J. Huang. http://www.bptrends.com/publicationfiles/01-05 etom and itil - huang.pdf, 2005.

[8] C. Bartolini and M. Sallé. Business driven prioritization of service incidents. (unpublished), 2005.

[9] J. Hellerstein, K. Katircioglu, and M. Surendra. A framework for applying inventory control to capacity management for utility computing. In *Proceedings of the IXth IFIP/IEEE International Symposium on Integrated Network Management IM'2005*, pages 237–250. IEEE, 2005.

[10] M. Sallé and C. Bartolini. Management by contact. In *Proceedings of 8th Network Operations and Management Symposium NOMS 2004*. IEEE/IFIP, 2004.

[11] R. Axelrod. *The Evolution of Co-operation*. Penguin Books, 1990 (1984).

[12] J. Sauvé et al. Sla design from a business perspective. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775*.

[13] G.B. Rodosek. Quality aspects in it service management. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 82, 2002.

[14] D. Daly, G. Kar, and W. Sanders. Modeling of service-level agreements for composed services. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 4, 2002.

[15] D. Trastour, C. Bartolini, and C. Priest. Semantic web support for the business-to-business e-commerce lifecycle, 2002.

[16] A. Daskalopulu and M. Sergot. The representation of legal contracts. *AI & Society*, 11:6–17, 1997.

[17] J.D. Carrillo and M. Dewatripont. Promises, promises. Technical Report 172782000000000058, UCLA Department of Economics, Levines's Bibliography.

[18] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. Summary of the stability of service level agreements. In *Proceedings of International Policy Workshop 2005*.

[19] Mark Burgess. An approach to policy based on autonomy and voluntary cooperation. *Lecture Notes on Computer Science*, 3775:97–108, 2005.

[20] M. Burgess and S. Fagernes. Pervasive computing management i: A model of network policy with local autonomy. *IEEE eTransactions on Network and Service Management*, page (submitted).

[21] M. Burgess and S. Fagernes. Pervasive computing management ii: Voluntary cooperation. *IEEE eTransactions on Network and Service Management*, page (submitted).

[22] M. Burgess. *Handbook of Network and System Administration*, chapter System Administration and the Scientific Method. Elsevier, 2007.

[23] R.B. Cooper. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Theory. Elsevier, 1990.

[24] J. Walrand. *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter Queueing Networks. Elsevier, 1990.

[25] J.P Bouchard and M. Potters. *The Theory of Finanical Risks*. Cambridge University Press, Cambridge, 2000.

[26] U.S. Nuclear Regulatory Commission NRC. *Fault Tree Handbook*. NUREG-0492, Springfield, 1981.

[27] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.

[28] M. Steinder and A. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 53:165, 2003.

[29] R. Apthorpe. A probabilistic approach to estimating computer system reliability. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 31, 2001.

[30] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. On the stability of service level agreements. *IEEE eTransactions on Network and System Management*, submitted 2005.

[31] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.

[32] J.L. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury. *Feedback Control of Computing Systems*. IEEE Press/Wiley Interscience, 2004.

[33] S. Axsäter. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*. Elsevier, 1993.

[34] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.

[35] D.A. Menascé and V.A.F. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prenctice Hall, 2000.

[36] G. Canright and K. Engø-Monsen. *Handbook of Network and System Administration*, chapter Some Aspects of Network Analysis and Graph Theory. Elsevier, 2007.

[37] M. Burgess and G. Canright. Scaling behaviour of peer configuration in logically ad hoc networks. *IEEE eTransactions on Network and Service Management*, 1:1, 2004.

[38] A. Bouch and M.A. Sasse. It ain't what you charge, it's the way that you do it: A user perspective of network qos and pricing. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 639, 1999.

[39] J. Topping. *Errors of Observation and their Treatment*. Chapman and Hall, 1972.

[40] R.H. Dieck. *Measurement and Uncertainty (Methods and Applications)*. Instrument, Systems and Automation Society, third edition edition, 2002.

[41] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, **2**:333, 1994.

[42] D. Verma et al. Policy based sla management in enterprise networks. In *Policy Workshop 2001*. Springer Verlag, 2001.

[43] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.

[44] A. Sahai et al. Automated sla monitoring for web services. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 28, 2002.

[45] P. Flegkas et al. Design and implementation of a policy-based resource management architecture. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 215, 2003.

[46] M. Debusmann and A. Keller. Sla-driven management of distributed systems using the common information model. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 563, 2003.

[47] Y. Diao, J.L. Hellerstein, and S. Parekh. Optimizing quality of service using fuzzy control. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 42, 2002.

[48] M. Burgess. Computer immunology. *Proceedings of the Twelth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 283, 1998.

[49] M. Burgess. Two dimensional time-series for anomaly detection and regulation in adaptive systems. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, LNCS 2506:169, 2002.

[50] Y. Diao et al. Generic on-line discovery of quantitative models for service level management. In *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 158, 2003.

[51] Arun V. Holden, editor. *Chaos*. Manchester University Press, 1986.

[52] H.L. Lee and S. Nahmias. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, chapter Single Product, Single Location Models. Elsevier, 1993.

[53] M. Burgess and T. Reitan. A risk analysis of disk backup or repository maintenance. *Science of Computer Programming*, (to appear), 2005.

[54] M. Burgess. Probabilistic anomaly detection in distributed computer networks. *Science of Computer Programming*, page (To appear), 2005.

Introducing CPU Time as a Scarce Resource in P2P Systems to Achieve Fair Use in a Distributed DNS

Thomas Bocek[1], David Hausheer[1], Reinhard Riedl[2], Burkhard Stiller[1,3]
[1]Communication Systems Group, Department of Informatics IFI, University of Zurich, Switzerland
[2]Information Systems Group, Department of Informatics IFI, University of Zurich, Switzerland
[3]Computer Engineering and Networks Laboratory TIK, ETH Zürich, Switzerland
E-Mail: [bocek|hausheer|riedl|stiller]@ifi.unizh.ch

Abstract - **Peer-to-peer (P2P) systems are flexible, robust, and self-organizing resource sharing infrastructures which are typically designed in a fully decentralized manner. However, a key problem of such systems are peers overusing a resource. This paper presents a fully decentralized scheme to achieve fair use in P2P systems, which does not require a priori information about a peer. The approach developed is based on a scarce resource trading scheme (SRTCPU), which utilizes CPU time as a form of payment. SRTCPU provides an incentive to offer CPU time in return of consuming a scarce resource. A distributed DNS has been implemented as an example application that uses SRTCPU.**

I. Introduction

A P2P system is fault-tolerant, robust and usually does not require any special administrative arrangements [3]. Currently, several P2P infrastructures exist, which can be used to implement large-scale applications, *e.g.*, CAN [19], Chord [23], Pastry [20], or Tapestry [25]. Applications based on these P2P systems may not only suffer from malicious nodes that can stop an application to operate by deploying a sibyl attack [11] or by pseudospoofing [10], but may also suffer from overuse of resources they do offer [8].

A node in a P2P system has typically a limited amount of resources that it may share with other nodes, *e.g.*, bandwidth, storage space, or central processing unit (CPU) power. However, if these resources are overused by other nodes, an unbalanced load is created such that nodes, which actively contribute to the network, are punished [14]. To reach a fair allocation of resources every node should only be allowed to use the amount of resources it provides. Three different accounting concepts can be applied to achieve such a fair allocation:

Centralized accounting can account for resource usage of every node in a P2P system. However, a central element is not a good option as the specific flexibility and robustness properties introduced by a decentralized P2P system would be weakened.

Decentralized collection of information about a node's behavior is an accounting concept that requires cooperation. Nodes collect information about resource usage and behavior of other nodes and provide other nodes with this information. Thus, decentralized collection of information is cooperative because the information is provided partly by third party nodes.

Decentralized resource trading allows nodes to be independent from other nodes. Resource trading denotes the concept that a node receives a resource in return for providing a resource [7]. Resource trading is fully decentralized, as a node can trade with its own resources and can decide on its own whether or not to provide resources. Thus, the node is independent of other nodes.

In this paper, a fully decentralized resource trading scheme is developed. The scheme prevents overuse of resources in a P2P system by introducing a payment for services, which is based on calculations using CPU time. It maintains a self-regulated, self-organized, and sustainable resource exchange targeted at a fair resource allocation. This approach is termed SRTCPU (scare resource trading with CPU time) [5]. The scheme proposed is evaluated using a name service as an application example. Thus, a Distributed Domain Name System (DDNS) prototype has been built that overcomes the resource overuse problem by applying SRTCPU. An analysis of this DDNS has been performed demonstrating that SRTCPU can be used in an integrated fashion with an application. The implementation and evaluation focuses on DDNS, where SRTCPU is a key component of it.

A first approach in developing a name service using P2P networks was developed in [8]. The key problem described there is the problem of *insertion denial of service*, which is considered in the context of this paper as a resource overuse. This problem deals with the possibility of a massive insertion of name value pairs. A node reserving every possible name combination in an endless loop could induce a stop in the network operation as new names can no longer be inserted. In contrast, in the traditional non-distributed DNS it is not possible to reserve such a great number of name combinations unless a huge amount of money is spent, since every name has a price to be paid.

Therefore, SRTCPU is proposed as a payment for name insertions into a DDNS. Unlike crypto puzzles, which determine a specific type of calculation in order to keep the CPU busy, SRTCPU can use the CPU for any type of calculation, *e.g.*, a useful calculation. This enables to contribute CPU time to a grid or other tasks which are in the need of computational power, *e.g.*, for solving large-scale computation problems, such as in SETI@home [21].

SRTCPU achieves a fair use, because a node can only use as much resources as an average node does, which means that

This work has been performed partially in the framework of EMANICS (FP6-2004-IST-4-026854-NoE).

a node cannot overuse a given resource. This does not completely compare to freeriding, since SRTCPU does not directly enforce that a node using resources has to provide resources as well. However, SRTCPU prevents freeriding indirectly. If a node *a* has been identified as malicious (*e.g.,* a freerider) by node *b*, then the provided resources from node *a* in order to use resources on node *b* are lost. The following example illustrates the difference between freeriding and fair use achieved by SRTCPU: *E.g.*, pure freeriding can be described as a user traveling by train without paying a fee. In contrast, with SRTCPU users potentially pay 5 cents for a single train ticket. If the price is too low, everyone may be traveling by train, thus, the train will become overcrowded and the price will rise to 5 dollar.

The remainder of this paper is organized as follows: While Section II . compares related work, Section III . defines the design of SRTCPU. Section IV . discusses the implementation of the distributed DNS prototype and validates requirements. Finally, Section V . summarizes and discusses future work.

II. Related Work

A review of related work has revealed a number of different approaches tackling the problem of resource overuse. Key characteristics taken into account cover the level of decentralization and the dependency on other nodes. These characteristics clearly distinguish the three concepts. Consequently, related work is discussed and compared according to the following dimensions:

- **Existence of a central element**. This indicates whether a central element is present or not.
- **Identification of fair resource usage based on local data.** Trust is good, control is better. A decision taken based on local data judging if a node is overusing a resource is always better then to trust other, potentially malicious nodes.
- **Efficient resource usage**. Counting back a hash is an exhaustive calculation. CPU time cannot be contributed to calculate anything else, *e.g.*, SETI@home [21].
- **Resource symmetry.** Resource symmetry denotes the trading with the same kind of resources, for example bandwidth in exchange for bandwidth.

For further information on related work, refer to [5].

A. Comparative Dimensions

Distributed systems applying decentralized or centralized schemes do show different behaviors. Thus, a comparative study has been performed. The notion of cooperation has been introduced to denote a high degree of dependency. In cooperative systems, *e.g.*, transitive trust management ([1] [9] [12] [15]), a node is dependent on more nodes than in independent systems. For example in tit-for-tat (TFT), a node can make a decision about sharing a resource based on the interaction between itself and another node. Examples for deploying a central element are: DNS, SETI@home [21], PPay [24], or A4C [22].

Resource trading can be divided into symmetric and asymmetric resource trading. Symmetric resource trading happens when the same kind of resource is traded between two nodes, *e.g.*, if one node requests *x* data sets from a second node, the first node has to provide the second node with *x* data sets, too. Bittorrent [6] is an example of symmetric resource trading with bandwidth taking place. However, symmetric resource trading only works, if the first node is interested in the resource of the second node

If symmetric resource trading is not possible another resource available for trading has to be found. Asymmetric resource trading happens when one type of resource is traded for a different type of resource, *e.g.*, storage space in return for CPU power. HashCash [2] and SRTCPU deploy an asymmetric resource trading scheme. Asymmetric Resource Trading implies dealing with exchange rates for trading different resources with different values. However, in a decentralized system, no supervisor or resource allocator is present. Therefore, decentralized resource trading has to happen in a completely self-regulated and self-organized way.

B. Comparison

With respect to SRTCPU, TFT and hashcash, no central elements are necessary and they are not dependent on data from other nodes for verifying fair resource usage. In addition, with SRTCPU CPU time can be used for arbitrary calculations. In contrast, hashcash keeps the CPU busy with a very specific computational task without any freedom of choosing an arbitrary calculation. This can be considered as a waste of CPU time. Finally, in contrast to TFT, no resource symmetry is required for SRTCPU. Thus, SRTCPU defines an optimal approach to prevent the overuse of a resource in a fully decentralized system. The full comparison of approaches is shown in Table , where "+" indicates the presence of the characteristic, while "-" indicates its absence.

TABLE I
Comparison of different approaches for resource trading

	No central element	Identification of fair resource usage based on local data	Efficient resource usage	No resource symmetry required
Tit-for-tat	+	+	+	-
Trust Management	-[a]	-	+	+
Hashcash [2]	+	+	-	+
Centralized Accounting	-	-	+	+
PeerMint [13]	+	-	+	+
SRTCPU	+	+	+	+

a. Trust Management has to limit the number of possible identities of a client. [11] show that without resource trading, a limitation is only possible with a central element.

III. Example-driven Design

Based on the investigation of related work, the key requirements for SRTCPU have been collected. The detailed design of SRTCPU includes the set of mechanisms in charge of sending computational tasks. Those need CPU time to be solved, basically in exchange for another resource to become as scarce as the CPU time. The basic design of such a use of CPU time includes a process to chose randomly a third party acting as a task provider.

A. Technical Prerequisites and Requirements

A node ID identifies a node, and SRTCPU relies on the fact that a node ID cannot be freely chosen. A prevention of arbitrary node ID selection can, *e.g.*, be achieved by calculating the node ID based on the IP address [23]. A second possibility is to use a central element, as proposed by [11] to assign node IDs.

The following 7 requirements have been defined to enable the design of a fully decentralized asymmetric resource trading scheme which will meet the key goals of a distributed system: efficiency and scalability with respect to the number of resources being dealt by and the number of users utilizing the scheme developed:

- Fair. Provide a fair allocation of resources, i.e. prevent resource overuse.
- Fully decentralized. No central elements should be present.
- Load balanced. A node should not become a bottleneck.
- Fault tolerant. A failure of nodes should not affect the service.
- Self-organized. The scheme is able to adapt to changes in the network.
- Efficient. The scheme should consume as little resources as possible.
- Trustworthy. In order to reduce the risk of attacks, minimal trust relations should be deployed.

B. Main Challenge

The main challenge of the architectural design is to ensure that a node has calculated a task by itself. Without this assurance the node could re-label the task and send it as a task provider to another node. To prevent such behavior a checksum of what has been calculated is being made. The same task from the same task provider results in the same checksum. The checksum is calculated using the instructions that have been performed and the node ID of the node that has commissioned the task. A malicious node that relabels a task and sends it back as a new task will receive a different checksum from that relabeled task. The node which received the relabeled task will use the node ID from the malicious node to calculate the checksum. As stated in the technical prerequisites, a node ID cannot be freely chosen and other nodes can verify that. As a consequence, a malicious node cannot fake its node ID without being detected.

C. SRTCPU Design

The basic design of SRTCPU is depicted in Figure 1. Three participants can be identified: A task provider that has a calculation to be outsourced, a requesting node (node 2) and a node that provides a resource (node 1). The number of requesting nodes is denoted *r*, the percentage of malicious nodes is *m*.

The communication starts with node 2 requesting a resource (1). In order to get that resource node 2 must be willing to provide CPU time for that resource. Node 1 sends this request to the task provider (2). Node 2 receives and calculates the task (3) which was requested from node 1. During task calculation, a checksum is generated which includes the node ID of node 1 and the instructions that have been executed during the task. The result and the checksum are sent back to the task provider (4). Node 1 receives the checksum from node 2 and the task provider (5), (6). This allows node 1 to determine if the task provider has received the result. If the checksum is not received from the task provider, an error has occurred either at node 2 or the task provider. Node 1 will mark both nodes in a local list as potentially malicious. If a certain threshold is reached as a node continues to behave incorrectly, that node will not be considered anymore as a trading partner in future.

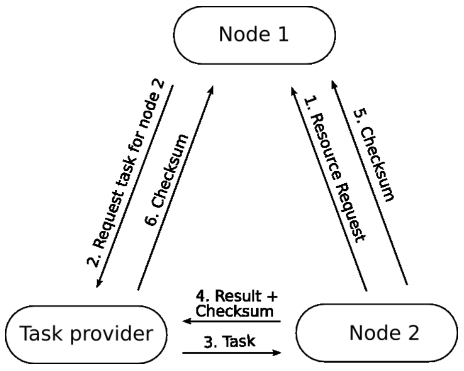

Figure 1. Basic design of SRTCPU with three participants (r=1, m=0).

A task provider can announce itself to any node. The task provider that announces itself to a node does also have to compute a task to get listed. A listed task provider is then randomly selected to provide a computational task.

A design with three participants and *r=1* is not feasible if a malicious node may send back arbitrary data and checksums. Therefore, more nodes (*r>1*) have to be introduced. Figure 2 shows the extended design with *r=2*. Node 4 has been introduced and is calculating the same task as node 3. It is obvious that a task has to be deterministic. Node 1 receives *r* checksums that have to be equal. If they are not, one node has made a mistake and both nodes are marked as potentially malicious. When the number of mistakes reaches a certain threshold, a malicious node is detected. Note, that this scheme only works when *m<0.5*.

Node 1 is also able to send a computational task, if no task provider is available or *r=1* with *m>0*. The task looks similar to hashcash, calculating back a hash, i.e. node 1 sends node 2 a hash value of an initial value. Node 2 has to calculate back a possible initial value for that hash value.

It is also possible to have *r>2*. With more checksums sent back, node 1 can determine faster if a node is behaving maliciously. However, with more nodes, more overhead is created.

The number of malicious nodes may vary from none to any number. Through parameter *r* and the type of task (crypto puzzle or arbitrary calculation), the impact of malicious nodes can be reduced or even eliminated. When using crypto puzzles the scheme can tolerate any number of malicious nodes independent of *r*, because the result of the task can be verified by the node that has sent the task. When using arbitrary calculations, with *r>1* a certain amount of malicious nodes can be present, because the checksum can be compared and, thus, malicious nodes can be detected and ignored. By varying the parameter *r* and the type of task, SRTCPU can be adapted to a certain number of malicious nodes trying to overuse a resource.

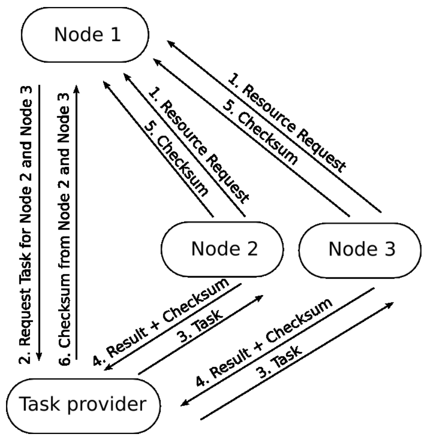

Figure 2. Extended design of SRTCPU with four participants. 2 nodes (node 2,node 3) have received the same task (r=2)

D. Self-Regulating Task Difficulty

With SRTCPU a resource is paid by calculating a task. The amount of payment is defined by the difficulty of the task which is an important factor in making a resource scarce. If the difficulty is set too low, overuse will be the result, if it is set too high, underuse will happen. Along with every checksum sent back, the number of processed instructions is sent to indicate if more computation is needed. Node 1 knows how much node 2 and node 3 have calculated by indicating the number of processed instructions that has been returned to node 1. Node 1 now needs to determine the average CPU time per inserted name value pair to keep the scheme self-regulated. This way, the scheme keeps track of an increase in CPU power over time [18].

Node 1 can determine the average CPU time with a benchmark on itself, under the assumption of having an average CPU. A tolerance factor will be applied to allow for a wider range of different CPU power. The tolerance factor determines how homogeneous in terms of CPU processing speed the system can be. This is an important factor because not every node has the same CPU power. Small and embedded devices have usually lower CPU power. SRTCPU rather focuses on homogeneous systems where a similar amount of resources are available on every node.

To calculate the average CPU time per inserted name, node 1 needs to know the average amount of data stored in the P2P system. Again, the node assumes that its stored data is the average. With the average CPU time and the average stored data, node 1 can decide if node 2 and node 3 have calculated enough to be allowed to store a name on node 1. Hence, the system is self-regulated.

IV. Implementation and Evaluation

DDNS has been implemented to validate the feasibility of a decentralized name service using SRTCPU. SRTCPU comes in place when a name is stored. With every store command, SRTCPU provides a task for the node that wants to store a data record.

The calculations of the task are executed in a virtual machine which performs only mathematical tasks. For the sake of simplicity, a prototypical MathVM [4] has been implemented in Java. The MathVM can update the checksum on every processed instruction. It uses its own language and has an assembler that translates the source code into instructions (opcodes). A task description in the MathVM has a fixed size of several Kilobyte. Only one task description representing a crypto puzzle has been implemented for SRTCPU. The prototype does not take into account any malicious nodes. DDNS is based on a DHT with an XOR metric [16] with an adaptation of the DDNS message format. DHTs provide a good performance for lookup, i.e. in *O(log n)*, where *n* is the number of nodes. The prototype has been tested with 1000 nodes to validate that the search scales gracefully [4].

A. DDNS Efficiency Analysis

The two most important commands of DDNS are *get* and *store*. A *get* is a data lookup that takes *O(log n)* in a structured overlay network [16]. A *store* performs first a lookup in *O(log n),* in order to find the node where to store the data. For the detailed communication protocol, please refer to [4].

Let *n* be the number of participating nodes. It is assumed that the number of queries scales with *O(1)*, which means that the number of queries a user performs are constant over time. The analysis of the *store* and *get* commands shows in the following that the message complexity remains *O(log n)* as in the Kademlia network [16].

The *get* command has been modified for SRTCPU to obtain always more than one result, but this does not affect *O(log n).* Based on the fact that the number of queries *p(n)* is *O(1)*, the overall number of messages which equals to *p(n)* * *O(log n)* is also *O(log n)*. The *store* command has been modified with the SRTCPU mechanism. First, every *store* performs a *get* to check if the name exists. For every *store* the message complexity is *p(n)+(p(n) * O(log n))*. Again *p(n)* is *O(1)* and the efficiency is *O(log n)*. Then the tasks and the results are being exchanged without any further lookup.

Even if *p(n)* would be *O(log n)*, the complexity of the store command would still be $O(\log n)^2$. However, with *p(n) = O(n)* the complexity would be *O(n * log n)*, which is too high. On the other hand, with the same situation that *p(n) = O(n),* the complexity of the DNS would be *O(n) * O(1) = O(n)*, thus the scheme would still not scale.

As a result of this analysis, all modifications do not exceed *O(log n)* and the lookup remains scalable. Nevertheless, some optimizations could boost the performance. One optimization is clustering as done in SHARK [17]: The namespace can be divided into parts. A part is built by grouping similar nodes. Similar in this context means that nodes searching the same names are similar. Let *p(n)* be the amount of nodes that searches within its parts and *q(n)* be the amount of nodes that searches in other parts. Having constant part sizes and growing numbers of parts, a *get* command in such a part is *O(1)*. The *get* command for searching in- and outside of a part is *p * O(1) + q * O(log n)*. The complexity remains *O(log n)*, but if a lot of searches are within a part, the *get* command becomes faster. The second optimization is caching, where the same situation arises. When caching a lot of entries the *get* command becomes faster, but there will also be uncached names, consequently the system remains *O(log n)*.

B. *DDNS Experiments*

The prototype implemented has been tested and several simulation experiments with up to 100 nodes per machine have been performed. 10 machines have been used to simulate a network.

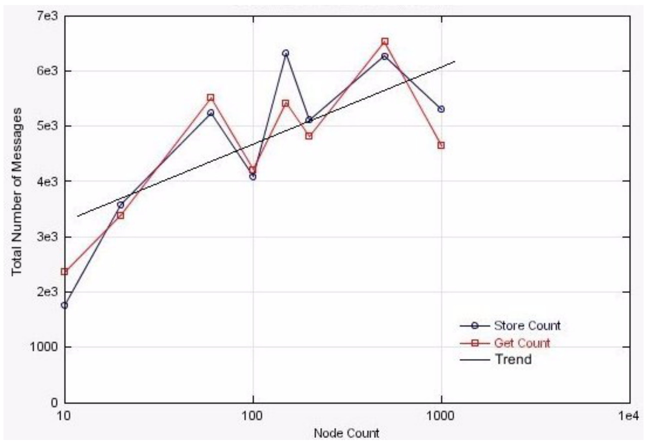

Figure 3. Simulation experiments of the prototype with SRTCPU as a part of DDNS.

These experiments have confirmed the results of the efficiency analysis. The simulation has been performed using DDNS as an example application that uses SRTCPU. To enable SRTCPU in DDNS the *store* command has been modified. With each *store* command, a request for a task is sent. A reply to a store command includes the task and after calculating the task, the result is being sent back. The result for the number of messages in total for *get* and *store* are depicted in Figure 3. This figure indicates that the communication cost grows with *O(log n)*. In the experiment 15 names were stored and 100 *get* requests were made. The *get* requests were made once and the *store* commands were repeatedly called. After all names had been stored and 100 requests had been made, the run was aborted. The numbers of nodes tested were 10, 20, 60, 100, 150, 200, 500, 1000. In each step, 2 runs were made and the mean was calculated. A node could fail with a probability of 10%. Malicious nodes were not considered (*m=0*), no task providers were present, and *r=1*.

Due to the implementation of the prototype with just one thread per communication channel, only a small number of nodes (app. 150 nodes) could be tested per machine.

C. *Validation and Discussion*

Key requirements discussed above have been met.

- Fair. Fairness is achieved since everyone has to pay about the same amount of CPU time for a resource. The amount of CPU time to be paid is self-organizing. Resource overuse is limited through the introduction of SRTCPU offering the possibility of trading different types of resources. In DDNS, a name value pair is traded for CPU time.
- Fully decentralized. SRTCPU is fully decentralized. No central element is necessary to account for resource usage.
- Load balanced and fault tolerant. Due to redundant storage a failure does not affect the scheme and the load is balanced. Additionally, a caching mechanism in DDNS could optimize the balance especially with popular names.
- Self-organized. With the use of CPU time to prevent overuse of a resource, every node can decide how much to charge for its resource. A central element is not necessary and a self-regulating task difficulty establishes a balanced charge of CPU time with respect to increasing processing power (Moors' law [18]) over time. Therefore, even when the resources change over time the system itself remains self-organized.
- Efficient. Although CPU time has to be spent, SRTCPU adds a benefit over a crypto puzzle by making the CPU time available for other calculations as well.
- Trustworthy. Trust can be measured based on local data. Thus, minimal trust relations have been achieved.

In SRTCPU, malicious nodes can either be detected by introducing redundancy and comparing the results of the calculations from other nodes, or by sending crypto puzzles as computational task. Malicious task providers can also be detected based on redundancy, i.e. nodes can randomly pick a task provider. Information about unexpected behavior of task providers and nodes will be collected and evaluated. If a certain threshold of unexpected behavior is reached, the node or task provider is considered as malicious. The number of task providers can also be limited by using SRTCPU when a new task provider joins the system, i.e. to get listed on a node, a task provider would have to pay with CPU time. The type of calculation that a task provider wants to have calculated is not important. To protect against misuse, an endless loop in a calculation will result in a time-out.

A problem arises when the network is very heterogeneous, i.e. with a lot of different CPU capabilities. [11] states that *large-scale distributed systems are inevitably heterogeneous.* Especially embedded and small devices have usually fewer resources to allocate. However, if a participation in a system requires certain resources, the problem of heterogeneity is less relevant. If a node wants to join the network it has to make sure that it has enough resources. If there are not sufficient resources, the node cannot join and has to be upgraded, *e.g.*, with more storage space or a faster processor. The result is that every SRTCPU participant will have a similar amount of resources available.

Efficiency is limited by the possible presence of malicious nodes and the absence of nodes for the verification of a computational task. Overhead is created as SRTCPU has to send a task to multiple nodes to verify the result. Therefore, the calculation is not as efficient as in a supervised system. A fallback strategy similar to hashcash is used when just one node requests a resource. In this case, the CPU time cannot be contributed to calculate anything else.

The bandwidth capacity of a node may become a bottleneck, if the task description is large. In the DDNS prototype, the task description is limited to a fixed size. Without this limitation, the required bandwidth would be considered as part of the payment.

It is not possible to guarantee uniqueness of names in a fully decentralized system. However, the ambiguities in DDNS may exist only for a limited time. If two names are inserted at the same time but from different peers, the name propagated faster will prevail [4].

V. Summary and Future Work

Introducing CPU time as a scare resource in a P2P system can make other resources as scare as the CPU time by binding these two resources together. As an example application of SRTCPU, DDNS has been implemented. DDNS is an architectural concept of a decentralized name service based on a P2P network. It implements SRTCPU to protect the critical resource *storage space for name value pairs*. The names can be retrieved in *O(log n)* due to the underlying DHT. With the introduction of task providers, one can use the CPU time, resulting from name insertions, for any task. It can, *e.g.*, be used in a grid, to solve large-scale computation problems similar to SETI@home. Contrary to the altruistic donation of CPU time to large-scale computation communities, DDNS creates an incentive to provide CPU time to a grid. Thus, the feasibility of DDNS has been validated with a prototypical implementation.

Future approaches may be based on different resources, other than CPU time. For example, a decentralized web page system with a name service could have storage space and bandwidth as the scarce resource. In order to store the name of a web page, storage space would have to be provided to other nodes.

To achieve a more detailed view of the impact of malicious nodes and to retrieve additional data from the simulation, simulation experiments are prepared to be made on a group of machines, mainly ba varying the parameter *r* and changing the number of malicious nodes.

VI. References

[1] K. Aberer, Z. Despotovic. *Managing trust in a peer-2-peer information system*. H. Paques, L. Liu, and D. Grossman (Edts): 10th International Conference on Information and Knowledge Management (CIKM'01), pp 310 — 317, New York, U.S.A., November 2001.

[2] A. Back. *Hash Cash — A denial of service counter-measure*, URL: http://www.hashcash.org/papers/hashcash.pdf, August 2002.

[3] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. *Looking up data in P2P systems*. Communications of the ACM, 46(2):43 — 48, February 2003.

[4] T. Bocek, *Feasibility, Pros and Cons for Distributed DNS,* Master Thesis, IFI, University of Zurich, May 2004.

[5] T. Bocek, D. Hausheer, R. Riedl, B. Stiller, *Introducing CPU Time as a Scarce Resource in P2P Systems*, University of Zurich, IFI, Technical Report No. ifi-2006.01, January 2006.

[6] B. Cohen. *Incentives Build Robustness in BitTorrent.* 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, California, U.S.A., June 2003.

[7] B. F. Cooper, H. Garcia-Molina, *Peer-to-Peer Resource Trading in a Reliable Distributed System*, Lecture Notes in Computer Science, Volume 2429, Springer, Berlin, pp 319 — 327, January 2002.

[8] R. Cox, A. Muthitacharoen, R. Morris, *Serving DNS using a peer-to-peer lookup service*, 1st International Workshop on Peer-to-Peer Systems, Cambridge, Massachusetts, U.S.A., March, 2002.

[9] E. Damiani, De Capitani di Vimercati, S. Paraboschi, P. Samarati, F. Violante. *A reputation-based approach for choosing reliable resources in peer-to-peer networks*. 9th ACM Conference on Computer and Communications Security, Washington DC, U.S.A., pp 207 — 216, November 2002.

[10] R. Dingledine, M. Freedman and D. Molnar, "Accountability", *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, Chapter 16, pp 271 — 340, O'REILLY Press, 2001.

[11] J. R. Douceur. *The sybil attack.* 1st International Workshop on Peer-to-Peer Systems (IPTPS02), Cambridge, Massachusetts, U.S.A., March 2002.

[12] M. Feldman and K. Lai and I. Stoica and J. Chuang, *Robust Incentive Techniques for Peer-to-Peer Networks,* ACM Electronic Commerce, New York, U.S.A., pp. 102 — 111, May 2004.

[13] D. Hausheer, B. Stiller, *PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications,* IFIP Networking 2005, Ontario, Canada, May 2005.

[14] S. Kamvar, M. Schlosser, and H. Garcia-Molina. *Incentives for Combatting Freeriding on P2P Networks*. Euro-Par, Klagenfurt, Austria, June, 2003.

[15] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina. *The eigentrust algorithm for reputation management in p2p networks.* 12th International World Wide Web Conference (WWW), Budapest, Hungary, May 2003.

[16] P. Maymounkov, D. Mazieres. Kademlia: *A peer-to-peer information system based on the xor metric.* 1st International Workshop on Peer to Peer Systems (IPTPS'02), Cambridge, Massachusetts, U.S.A., March 2002.

[17] J. Mischke, B. Stiller: *Rich and Scalable Peer-to-Peer Search with SHARK*, Autonomic Computing Workshop — 5th Annual International Workshop on Active Middleware Services (AMS'03), Seattle, Washington, U.S.A., 2003.

[18] G. Moore: *Cramming more components onto integrated circuits*; Electronics, Vol. 38, No. 8, April 1965.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. *A scalable content addressable network.* ACM SIGCOMM'01, San Diego, California, U.S.A., pp 161 — 172, August 2001.

[20] A. Rowstron, P. Druschel. *Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems*. 18th IFIP/ACM Conference on Distributed Systems Platforms, Heidelberg, Germany, November 2001.

[21] SETI@home, URL: http://setiathome.ssl.berkeley.edu/, August 2005.

[22] B. Stiller, J. Fernandez, Hasan, P. Kurtansky, W. Lu, D.-J. Plas, B. Weyl, H. Ziemek, B. Bhushan: *Design of an Advanced A4C Framework,* Whitepaper, EU IST Project Daidalos, http://www.ist-daidalos.org/, August 2005.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. Chord: *A scalable peer-to-peer lookup service for internet applications*. ACM SIGCOMM'01, San Diego, California, U.S.A., pp 149 — 160, March 2001.

[24] B. Yang, H. Garcia-Molina. P*Pay: Micropayments for Peer-to-Peer Systems*. Technical Report, Stanford University, California, U.S.A., 2003.

[25] B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph. *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*, Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, California, U.S.A., April 2001.